Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática,
2009

Luís Miguel
Capela Gandarinho

Cluster computacional para processamento de reflexão sísmica 2D/3D

Luís Miguel Capela Gandarinho

Cluster computacional para processamento de reflexão sísmica 2D/3D

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia de Computadores e Telemática, realizada sob a orientação científica de António Guilherme Rocha Campos, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor Francisco José Curado Mendes Teixeira, do Departamento de Geociências da Universidade de Aveiro.

o júri / the jury

presidente / president	**António Rui de Oliveira e Silva Borges** Professor Associado da Universidade de Aveiro
vogais / examiners committee Arguente Externo	**Jorge Manuel Gomes Barbosa** Professor Auxiliar do Departamento de Engenharia Informática da FEUP
Orientadores	**António Guilherme Rocha Campos** Professor Auxiliar da Universidade de Aveiro
	Dr. Francisco José Curado Mendes Teixeira Departamento de Geociências da Universidade de Aveiro

agradecimentos / acknowledgements

Durante a realização desta tese houve pessoas que, de maneira directa ou indirecta, me acompanharam e a quem não poderia deixar de agradecer pois a sua ajuda foi fundamental:

Ao professor António Guilherme Campos, orientador desta tese, devo o profundo agradecimento pelo seu conhecimento e orientação, mas sobretudo pela sua valiosa contribuição nas indicações que tanto serviram para a boa realização deste trabalho. Agradeço todo o conhecimento que me foi transmitido, a disponibilidade e confiança que tornaram possível o desenvolvimento deste trabalho.

Ao Dr. Francisco Curado, também orientador desta tese, pelo acompanhamento e disponibilidade em todas as questões, que se mostraram fundamentais para que todo o trabalho fosse desenvolvido de uma maneira eficiente.

Em particular, ao professor Luís Menezes Pinheiro, director do Laboratório de Geologia Marinha do Departamento de Geociências de quem partiu a ideia deste trabalho e acolheu a sua realização, essencial ao desenvolvimento deste trabalho, pelos esclarecimentos, sugestões e incentivos indispensáveis para a realização da mesma.

Ao Dr. Victor Magalhães pela disponibilidade e colaboração na realização das experiências práticas desta tese.

Ao Dr. Dan Herold da *Parallel Geoscience*(empresa fornecedora do software SPW), pelo seu valioso contributo para o funcionamento do software, e pelas longas horas de trabalho que disponibilizou para ajudar a configurar correctamente o *cluster* computacional.

Ao Calvin Clark da Microsoft, pelos seus esclarecimentos do funcionamento do *Compute Cluster Server*, que se mostraram indispensáveis para que o *Cluster* funcionasse na perfeição.

Ao Centro de Informática da Universidade de Aveiro (CICUA) pela assistência prestada.

Ao Kamran Mustafa, com o qual trabalhei mais directamente, pela sua paciência e disponibilidade na realização dos testes e esclarecimentos sobre sísmica de reflexão 3D.

A toda a equipa do "geofmar"(Laboratório de Geologia Marinha do Departamento de Geociências) que sempre esteve presente, quer para partilha de conhecimentos, quer pela partilha dos momentos de descontracção, e por isso me fez sentir parte da equipa. Obrigado a todos, Ana Margarida, Catarina, Daniela, Kamran, Leonardo, Ronaldo, Tiago.

agradecimentos / acknowledgements

À família, em especial aos meus pais, que sempre me proporcionaram todas as condições e incentivaram, e ao meu irmão que sempre esteve presente ao longo do meu percurso académico. Sem o seu apoio e encorajamento, nunca teria sido possível chegar a este ponto.

À minha namorada, que me aturou em tantos momentos de má disposição e a todos os amigos e colegas da universidade de Aveiro com quem me cruzei nestes últimos anos.

Ao meu afilhado João que sempre esteve comigo nos momentos mais baixos e mais altos e à sua esposa Custódia pela disponibilidade e amizade.

Ao meu amigo Emanuel Fidalgo, pela cedência do computador que ajudou a testar a tese numa fase intermédias e ainda pela paciência e ajuda.

Aos meus amigos e colegas que sempre me acompanharam e encorajaram.

Agradeço ainda a outros amigos, muito especiais que me ajudaram a construir a pessoa que hoje sou.

Aos Sementes que sempre estiveram comigo e sempre me fizeram sorrir e a cada semana me dão um novo ânimo para continuar a trabalhar.

A estes e muitos outros que não citei o meu Obrigado.

Bem haja a todos.

Palavras-chave Cluster, Testes de Desempenho, Computadores Paralelos, Sísmica, Processamento Paralelo, Compute Cluster Server

Resumo

Tendo em vista o processamento de dados de sísmica de reflexão 3D, foi criado um *cluster* computacional usando *worstations* do Departamento de Geociências ligadas em rede Ethernet Gb, funcionando sob o sistema operativo *Windows Server 2003* com o *Compute Cluster Pack*. Apresenta-se uma descrição detalhada das operações associadas a cada etapa da implementação desse *cluster*. O seu funcionamento foi testado com diferentes processos utilizando um pacote de *software* comercial de processamento de dados sísmicos (SPW) preparado para tirar partido do processamento paralelo. Os testes permitiram comprovar e quantificar os ganhos de desempenho obtidos em termos das métricas habitualmente usadas para o efeito, nomeadamente *overhead, speedup, efficiency* e *scalability*. Em alguns processos de especial interesse prático, testados num *cluster* de 4 máquinas quad-core(16 processadores), atingiram-se ganhos da ordem de 11-12. Além disso a elevada eficiência atingida(da ordem dos 70-75), mostrou-se aproximadamente independente do número de processadores.

Keywords Cluster, Performance Tests, Parallel Computing, Seismic, Parallel Processing, Compute Cluster Server

Abstract Having regard to 3D seismic reflection data processing, was created a compute cluster using Department of Geosciences networked Gb Ethernet worstations, running under operating system Windows Server 2003 with Compute Cluster Pack. It presents a detailed description of the operations associated with each stage of the implementation of this cluster. Its operation was tested with different processes using a commercial software package for seismic data processing (SPW) prepared to take advantage of parallel processing. The tests allowed to prove and quantify the performance gains achieved in terms of the metrics commonly used for this purpose, particularly overhead, speedup, efficiency and scalability. In some special cases of practical interest, tested on a cluster of 4 quad-core machines (16 processors), reached up gains of about 11-12. Besides the high efficiency achieved (approximately 70-75), appeared to be nearly independent of the number of processors.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nos dias de hoje as exigências computacionais são maiores do que nunca, pois quanto maior for a capacidade de processamento, mais rápida se torna a realização de um trabalho. Imagine-se que existe uma determinada tarefa computacional realizada num tempo T com um único processador; em condições ideais um conjunto de n processadores poderá realizar a mesma tarefa num tempo $\frac{T}{n}$.

Face à tendência observada nas últimas décadas é previsível que em poucos anos, a mesma capacidade de processamento esteja disponível numa única máquina a um preço acessível[1], mas o mercado não espera por isso; existe uma pressão constante para a optimização dos recursos computacionais existentes - Daí a aposta num sistema de processamento paralelo.

Alguns dos problemas importantes a nível científico são tão complexos, que resolvê-los através de uma simulação numérica requer uma capacidade de processamento extraordinária. Estes problemas tão complexos pertencem a variadas áreas, nomeadamente:

- Meteorologia
- Geofísica/Geologia
- Química quântica.
- Cosmologia/Astrofísica.
- Biologia
- Medicina - Genética

Desde o ano 2000 têm vindo a ser criadas, de forma crescente, instalações de *clusters* de super-computadores. Um exemplo, é o *Earth Simulator* (Figura1.1), localizado no Japão. Neste momento conta com 640 nós de computação com oito processadores vectoriais e 16 GB de memória em cada nó, fazendo um total de 5120 processadores e 10 TB de memória.

[1] Lei de Moore

Figura 1.1: Centro de Computação do *Earth Simulator*.

Embora este *cluster* seja vocacionado para a Geofísica, processa também dados de outras áreas, como por exemplo a Genética.

Podemos encontrar esta computação de alta capacidade também em Portugal. A Computação **GRID** (a que alguns também chamam Computação em "Grelha") é uma tecnologia de computação distribuída que nasceu em 1994-95. A ideia é replicar para o processamento computacional, a filosofia e os princípios de funcionamento da *World Wide Web* (WWW) para a disponibilização de informação à escala mundial. A GRID é um projecto Europeu iniciado pelo **CERN** para o processamento de enormes quantidades de informação ligadas à física. Em Portugal, este projecto foi iniciado em 2001, com financiamento do Programa Quadro Europeu de Investigação e participação do Laboratório de Instrumentação e Física Experimental de Partículas (LIP) (Lisboa e Coimbra), das universidades do Porto e do Minho, do Centro de Física de Plasmas do Instituto Superior Técnico, do Instituto de Engenharia Electrónica e Telemática (**IEETA**) da Universidade de Aveiro, e da Universidade Lusíada (Famalicão). Esta rede europeia de computação integra actualmente em 41.000 CPUs e 5 PB de disco (5 milhões de GB).
A *INGRID* é a Iniciativa Nacional GRID portuguesa, lançada no dia 29 de Abril de 2006, em cumprimento do que ficou estabelecido no programa *Ligar Portugal*, definido em Julho de 2005 com o objectivo de orientar o desenvolvimento da Sociedade da Informação no país. A *INGRID* foi planeada, é acompanhada e parcialmente financiada pela Agência para a Sociedade do Conhecimento (UMIC), e é executada no que respeita a projectos de I D através de financiamentos concedidos pela Fundação para a Ciência e a Tecnologia (FCT).

Alguns exemplos da aplicação da **GRID** em Portugal são aplicações de controlo, previsão e simulação de cheias em bacias fluviais, de modelação de poluição atmosférica e de serviços de apoio à meteorologia, bem como uma aplicação de apoio à cirurgia vascular. Nesta última aplicação, partindo das imagens de TAC ou Ressonância Magnética de um paciente, o cirurgião pode simular o efeito de um *by-pass* e visualizar o fluxo sanguíneo, decidindo por tentativas sucessivas a melhor solução para a intervenção. A **INGRID** contava no final de 2008 com 1778 *CPUs* e 996 *TB* de memória.

Esta tese aborda um tipo de *cluster* com muito menor capacidade, um constituído por máquinas que servem não só para trabalho de processamento computacional, mas também

para trabalharem individualmente em trabalho normal de *Desktop* (PC). Obtém-se assim uma solução versátil e de baixo custo, com menor rentabilização do funcionamento das máquinas. Quando é necessário processar uma quantidade de dados razoável, ligam-se as máquinas ao *cluster*. Quando esta necessidade não surge, realizam a sua função normal de PC, ou seja, é fácil colocar e retirar máquinas do funcionamento do *cluster*. É importante referir também que a qualquer altura se podem adicionar de forma simples máquinas a este *cluster*, bastando para isso que obedeçam aos requisitos requeridos e que serão referidos adiante.

Assim, o presente trabalho propõe-se apresentar uma descrição detalhada das operações associadas a cada etapa da criação de um *Cluster* computacional de dados em *Windows Compute Cluster Server*, nomeadamente uma breve exposição sobre dados sísmicos, os passos de configuração do Sistema Operativo *Windows Compute Cluster Server* e do *software* de processamento associado ao processamento de dados de sísmica de reflexão, *Seismic Processing Workshop*. Depois destas configurações foram realizados testes de processamento de algumas linhas sísmicas que verificam o bom funcionamento do *Cluster*. Os resultados associados a esses testes serão apresentados no capítulo 5.

Assim sendo, pretende-se estudar não só o funcionamento de um *Cluster* em Windows, mas também o funcionamento do software usado para o processamento dos dados, fazendo uma breve abordagem aos tipos de dados processados e à forma como estes são armazenados. O trabalho foi consistiu portanto em colocar o *cluster* a funcionar e fazendo processamento de dados sísmicos na execução do *Seismic Processing Workshop*.
Os resultados relativos aos testes feitos e respectivas avaliações em termos das métricas adoptadas em processamento paralelo, nomeadamente.*speedup, efficiency* e *scalability*.

Capítulo 2

Aquisição e Processamento de Dados Sísmicos

Neste capítulo aborda-se o processamento de dados de Sísmica de Reflexão, que pertence à área de estudo da Geofísica. A sísmica de reflexão é um método de prospecção geofísica muito usado na investigação do interior da terra, em particular na indústria petrolífera. Este processo de aquisição de dados utiliza uma fonte geradora de ondas sonoras que se propagam e são reflectidas pelas várias camadas do subsolo. Essa resposta reflexiva é captada por dispositivos devidamente espaçados e colocados de forma a obtê-la nas melhores condições. A Figura 2.1 exemplifica o funcionamento deste método.

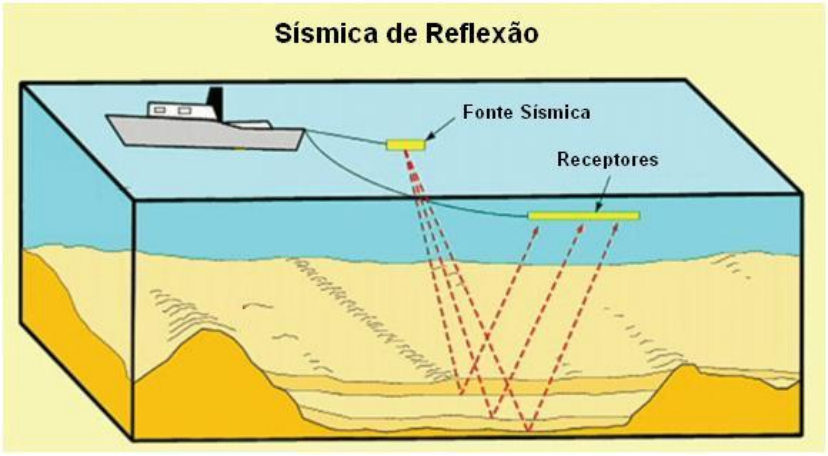

Figura 2.1: Método de Aquisição de Dados de Sísmica de Reflexão Marinha.[19]

Um sismo é um fenómeno de vibração brusca e passageira da superfície da Terra. O movimento é causado pela libertação rápida de grandes quantidades de energia sob a forma de ondas sísmicas. Tendo em conta que uma onda sísmica é uma onda que se propaga através da Terra, geralmente como consequência de um sismo, ou devido a uma explosão, verifica-se uma semelhança entre os sinais gerados por este método de reflexão sísmica e os sinais gerados por um sismo. A principal diferença entre as ondas criadas por este método de prospecção e as criadas por um terramoto é que as primeiras são induzidas artificialmente de uma maneira controlada e muito localizada, com parâmetros ajustados aos objectivos de

investigação propostos.
A partir do estudo do tempo de propagação das ondas sísmicas, o geofísico tem a oportunidade de explorar o subsolo sem necessidade de penetrar fisicamente as formações.

2.1 Aquisição de Dados

Há várias técnicas de geração artificial de ondas sísmicas. As fontes de energia mais comuns são:

- explosivos (dinamite);
- vibradores mecânicos;
- canhões de ar comprimido(*air guns*);

Ao nível dos receptores usam-se:

- **geofones**, em terra;
- **hidrofones**, na água.

Os geofones são muito usados para prospecção de petróleo em terra, enquanto que os hidrofones são utilizados na execução do reconhecimento das camadas do subsolo abaixo do fundo do mar e mais recentemente também na camada de água.
A generalidade dos dados utilizados neste trabalho foi adquirida no mar, utilizando "canhões"de ar comprimido, Figura 2.2) e um *streamer* de **hidrofones** (Figura 2.3).

Figura 2.2: Sistema de aquisição de dados sísmicos multicanal 3D.

Figura 2.3: *Air Gun* à esquerda e um hidrofone e respectiva *streamer* à direita.

Neste método, como é ilustrado na figura 2.2, são usados um ou vários canhões de ar comprimido, e uma cadeia (*streamer* na nomenclatura inglesa) de hidrofones, à ré do navio. Esta *streamer* pode ter vários quilómetros, compreendendo centenas de hidrofones. A técnica consiste em navegar na área que se quer analisar e ir disparando vários tiros em pontos diferentes. As ondas geradas por esses tiros vão ser propagadas e/ou reflectidas conforme a natureza do solo e a sua reflexão captada pelos vários hidrofones. É importante referir que os hidrofones têm que estar colocados a uma determinada distância abaixo do nível da superfície da água. A ideia da aquisição sísmica a 3D foi introduzida em 1972 pelo geofísico americano G.G. Walton. A partir daí os geofísicos de todo o mundo começaram a considerar o processo de aquisição, processamento e interpretação numa outra perspectiva.
A grande diferença do 2D para o 3D é relativa ao número de *streamers* de hidrofones. Em 2D apenas temos uma *streamer*, em 3D temos um conjunto de *streamers* de hidrofones na ré da embarcação, colocadas em paralelo. Desta forma, quando é feito um disparo, as reflexões são captadas por mais hidrofones, obtendo-se assim uma maior área coberta.
Ao contrário do método 2D, que fornece apenas informação em profundidade ao longo de uma linha recta, a sísmica 3D fornece um cubo de dados sísmicos relativas às 3 dimensões do espaço (X, Y e profundidade) relacionado com o tempo de atraso[1]. Estas dimensões são organizadas em *inlines* com a mesma direcção da faixa de aquisição, e em *crosslines* perpendiculares ao caminho de aquisição. Dependendo da qualidade dos dados nos volumes de sísmica 3D, o interprete é capaz de mapear eventos sísmicos ao longo da cadeia de aquisição e construir um bom modelo geológico. [10]

Com o decorrer do tempo, e sobretudo com os avanços tecnológicos, nomeadamente o aparecimento de computadores mais poderosos, capazes de processar e interpretar dados a 3D, a aquisição tridimensional começou a vulgarizar-se. Hoje conseguem obter-se mapas com uma resolução e qualidade muito elevada, como ilustra a figura 2.4.[1]

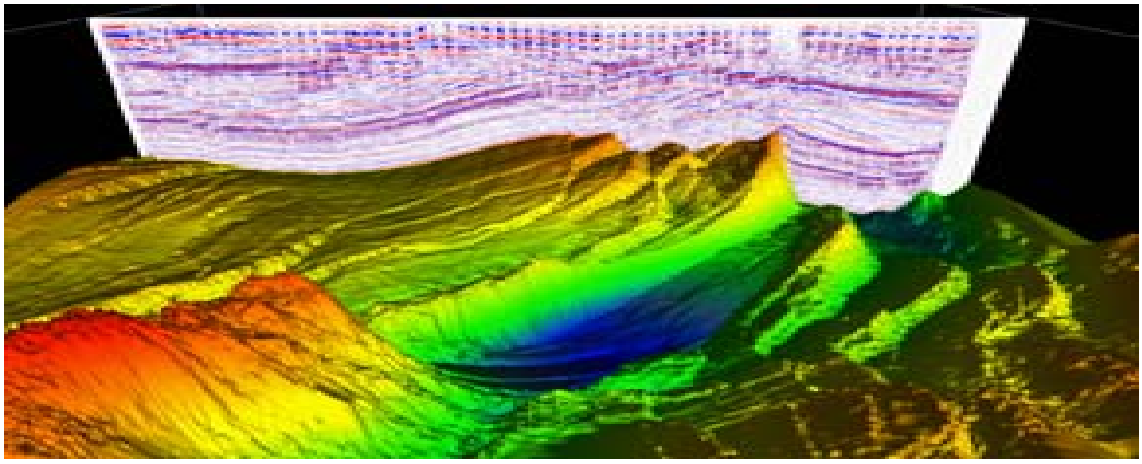

Figura 2.4: Resultados de Sísmica 3D.

Para que seja feita uma boa campanha de aquisição de dados sísmicos é imperativo que a geometria, o arranjo de fontes (*shots*) e receptores (*receivers*) registem o sinal sísmico com

[1]eixo X - horizontal na direcção da faixa de aquisição, eixo Y - horizontal perpendicular ao eixo X, eixo Z - profundidade.

a maior gama dinâmica(*dynamic range*) possível, sem distorção e reduzindo o ruído. Como a aquisição tridimensional é um método com uma grande área de cobertura, consegue-se de imediato melhorias na razão sinal/ruído e múltiplas atenuações, nomeadamente, sobre-amostragem da mesma reflexão. Adicionalmente, o resultado de uma densa grelha de amostragem e qualidade dos dados torna possível cartografar estruturas do subsolo, assim como formações geológicas.[2]

A aquisição dos dados explicada anteriormente produz ficheiros com uma quantidade de dados num formato ***raw***, ou seja, um formato de dados de aquisição não tratados com duas variantes: uma relativa à sequência temporal (*time-sequencial format*) e outra ao número de traços (*trace-sequencial format*). No *time-sequencial format* temos uma sequência de dados na qual a primeira amostra do primeiro canal é seguida pela primeira amostra do segundo canal e depois pela primeira do terceiro canal e assim sucessivamente até que todos os canais estejam colocados; de seguida são colocadas as segundas amostras de todos os canais pela mesma ordem e assim sucessivamente até que todas as amostras de todos os canais estejam colocadas. Em oposição, temos o *trace-sequencial format*, no qual todas as amostras do primeiro canal são colocadas, depois todas as do segundo e assim sucessivamente, até que termine o número de canais. Os dados podem ser colocados no formato matricial mostrado na tabela 2.1.

Tabela 2.1: Time-sequencial format vs. Trace-sequencial format

	Amostra 1	Amostra 2	Amostra 3	...	Amostra n
Canal 1	a1	a2	a3	...	an
Canal 2	b1	b2	b3	...	bn
Canal 3	c1	c2	c3	...	cn
...	...	...	...	...	...
Canal k	k1	k2	k3	...	kn

Desta forma o *time-sequencial format* seria dado pelas colunas e o *trace sequencial format* pelas linhas. Passa-se de um para outro para transposição da matriz.
Os dados de aquisição (dados de saída/ *output*) dos equipamentos de gravação digital estão, normalmente, num formato multiplexado. No entanto a maior parte do processamento de dados é feito em *trace-sequencial format*. A conversão entre estes dois formatos (*demultiplexing*) é um dos primeiros passos no processamento de dados e, normalmente, é parte de uma rotina de edição. Este tipo de conversão é dado pela transposta da matriz.

2.2 Processamento de Sinal (*Flow*)

Para que os dados sísmicos sejam interpretáveis, é necessário processar os registos sísmicos contidos nas bandas magnéticas de terra usando algoritmos sofisticados.
É só a partir do processamento de dados sísmicos que se consegue obter uma secção, ou imagem sísmica, que se pareça com uma secção geológica.
O volume de dados adquirido pode ser muito elevado; varia de acordo com o número de tiros (*traces*), e o número de canais registados, por sua vez relacionado com o número de receptores usados, a área de aquisição, o intervalo de amostragem e o tempo de captura.
Exemplo:

Se temos uma campanha de aquisição com:

- tempo de aquisição = 4000 s
- Intervalo de amostragem = 2000 us
- Número de tiros = 13794 traces
- Número de canais = 1

Como é apresentado na figura 2.5:

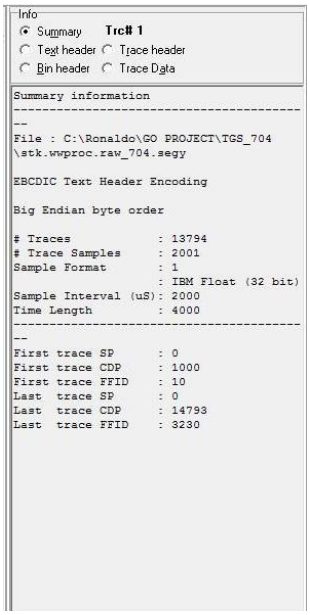

Figura 2.5: Informações de um ficheiro de dados sísmicos

Obtemos uma estimativa do volume de dados na ordem dos 107 MB.

O Volume é dado por:

$$\text{V(B)} = \frac{TimeLength(s)}{SampleInterval(s)} \text{ x Number of Chanels x Sample Format(B) x traces}$$

Para este exemplo foi usada uma linha muito pequena com uma quantidade relativamente pequena de informações. Tendo em conta que linhas adquiridas completas chegam a ter 10, 15 ou 20 vezes mais que esta verifica-se que se lidam com dados muito pesados para serem processados, até chegarem ao ponto em que é realmente possível analisá-los.

Para se passar de um registo de ondas sísmicas para uma secção geológica que reproduza um modelo realista são necessários 2 tipos de correcções, envolvendo cada uma delas uma série de processos: Correcção Dinâmica e Correcção Estática.[12]

Assim sendo, o tratamento de dados sísmicos consiste nos seguintes processos:

1.Aquisição sísmica.

2.Redução do Volume de Dados.

3.Correcções Geométricas.

4.Amplificação do Sinal.

Ao nível da aquisição da sísmica já foram descritos alguns métodos anteriormente. Relativamente aos outros passos, que se podem aplicar aos dados numa fase de pós-aquisição, serão descritos na Tabela 2.2 mais detalhadamente. Alguns destes processos irão ser testados no decorrer deste projecto para verificar a melhoria do tempo de processamento e quais deles conseguem uma melhor paralelização.

Tabela 2.2: Sequência Típica de Processamento Sísmico

PROCESSOS	**RESULTADOS DESEJADOS**
REDUÇÃO DE DADOS	
Desmultiplicação	Colocar os dados em formatos sequenciais de traços
Recuperação da Amplitude Real	Multiplicar os dados por códigos de ganhos binário
Edição	Eliminar os maus registos
Correlação	Para dados provenientes de vibradores
Processamento de ganho	Amplificação dos eventos profundos
CORRECÇÕES GEOMÉTRICAS	
CDP	Arranjo dos traços de acordo com o CDP
Correcção Estática	Correcção do Tempo Vertical
NMO	Correcção do Tempo Horizontal
AMPLIFICAÇÃO DO SINAL	
Muting	Eliminar Ruído de Alta Amplitude
Desconvolução	Conversão da forma dos impulsos de input e atenuação
Stack	da reverberação
Filtros	Atenuação do Ruído
Equalização	Atenuação do Ruído com frequências fora da banda do sinal
Conversão do Tempo em Profundidade	Amplificação dos eventos fracos relativos aos eventos fortes.

Para processar estes dados, tentando optimizar ao máximo a capacidade de processamento e aproveitando a paralelização dos dados é usada, nesta tese, uma combinação entre dois *softwares*: o Sistema Operativo *Windows Server 2003 x64*, com o pacote para criação de *Clusters*, *Compute Cluster Pack* e o software de processamento de sinal sísmico da *Parallel Geoscience*, *Seismic Processing Workshop*. No próximo capítulo serão abordados, de forma geral, os tipos de processamento paralelo.

Capítulo 3

Processamento Paralelo

3.1 Introdução

O processamento paralelo baseia-se na distribuição de trabalho por vários processadores interligados por um sistema de comunicação, de forma que todos cooperem na solução do problema e que o seu tempo de processamento seja reduzido.

Para abordar o processamento paralelo e da sua eficiência temos que ter em conta que quando tentamos tornar uma tarefa paralela, existem dados que são paralelizáveis, isto é, há dados que se podem dividir por vários processadores e ser processados individualmente e outros que não. Quanto maior for a capacidade de paralelização dos dados maior a eficiência do algoritmo e menor o tempo de processamento.

Essencialmente, a paralelização envolve dois aspectos: a **decomposição** e a **coordenação**. A Decomposição implica a divisão de um algoritmo em tarefas, que possam ser executadas concorrentemente de forma eficiente. A Coordenação por sua vez indica que deve obrigatoriamente, como o nome indica, existir algum tipo de coordenação, de tal forma que a contribuição dos processadores seja adequadamente coordenada e seja garantida uma evolução segura e eficiente do processo para a solução do problema.

O conceito de *Cluster* de computadores veio revolucionar o processamento paralelo. Essencialmente este consiste em ligar várias máquinas através de uma rede de forma a que estas comuniquem e partilhem as suas capacidades de processamento.

3.2 Computadores Paralelos

Os computadores paralelos podem ser divididos em três tipos:

- Computadores *Pipeline*.
- Computadores Vectoriais.
- Computadores Multi-processador.

Destes três tipos apenas será dada uma breve definição dos dois primeiros, visto que no caso desta tese o que será mais relevante será o terceiro tipo, pelo que, será dele que se dará mais ênfase.

3.2.1 Computadores *Pipeline*

Realizam processos de cálculo sobrepostos, explorando um paralelismo temporal, ou seja, permitem que o processador realize a busca de uma ou mais instruções além da próxima a ser executada. Estas instruções são colocadas numa fila de espera, dentro do processador, onde aguardam a sua vez para serem executadas. A técnica de *pipeline* é utilizada para acelerar a velocidade de operação do processador, uma vez que a próxima instrução a ser executada está normalmente armazenada dentro do mesmo e não precisa ser acedida através da memória, normalmente muito mais lenta que o processador. Este tipo de computadores é também referido como MISD (*multiple-instruction, single-data*).

3.2.2 Computadores Vectoriais

Os Computadores Vectoriais(*Array*, da nomenclatura inglesa), usam múltiplas unidades lógicas-aritméticas sincronizadas para conseguir paralelismo espacial. Estes são especialmente projectados para realizar operações vectoriais ou sobre matrizes. Pelo facto de executarem uma única instrução de cada vez, operando sobre um conjunto de dados simultâneos, são também chamados computadores SIMD (*Simple Instruction Multiple Data*). De referir ainda que este tipo de computadores pode variar em dois aspectos:

- O número de processadores pode ser fixo.
- A comunicação entre os processadores pode ser realizada via memória compartilhada, rede malha conectada, rede em pirâmide, hipercubo, entre outros.

3.2.3 Multi-processadores/Multi-computadores

No que respeita aos computadores que têm maior relevância nesta tese, baseiam-se em Multi-processadores. São sistemas compostos por vários processadores de capacidade comparável, que operam de maneira assíncrona. Todos os computadores têm memória local e outros recursos privados; eles podem comunicar com outros computadores através de memórias compartilhadas ou através de uma rede.

Neste ponto é necessário estabelecer uma diferença entre multi-processador e multi-computador. Por vezes as definições destes dois termos são confundidas.

Num nível muito básico, ambos têm o mesmo objectivo, ou seja, permitir a realização de operações concorrentes num determinado sistema. A grande diferença nota-se ao nível de cooperação e grau de partilha de recursos para a solução do problema.

Um Multi-computador consiste em vários computadores autónomos que comunicam entre si. Um Multi-processador é controlado por um único sistema operacional, que gere a interacção entre os processadores e os seus programas, nos níveis de processo e estrutura de dados. Neste tipo existe sincronismo entre processadores.

Os Multi-processadores podem ser classificados, de acordo com a sua forma de comunicação, de duas formas:

1.Multi-processadores fortemente acoplados.

2.Multi-processadores fracamente acoplados.

Multi-processadores Fortemente Acoplados

Nos Multi-processadores fortemente acoplados, tipicamente, a comunicação é realizada através de uma memória comum e compartilhada, e cada processador tem uma pequena memória local ou *buffer* de alta velocidade (cache). Este tipo de arquitectura é caracterizada pela elevada velocidade de comunicação entre os processadores, sendo que são necessários mecanismos de sincronização para evitar que endereços compartilhados sejam acedidos por mais que um processo em simultâneo.

Multi-processadores Fracamente Acoplados

Estes Multi-Computadores, muitas vezes referidos como *clusters*, são baseados em varias máquinas uni- ou multi-core conectadas através de uma rede comunicação de alta velocidade, nomeadamente ligações *Ethernet* a *Gigabit*. Um exemplo deste tipo de *Cluster* é o *Cluster Beowulf*. *Beowulf* são *clusters* de desempenho escalável baseados em *hardware*, colocados numa rede privada, com software de código aberto (Linux). O responsável pela manutenção do *cluster* pode melhorar o desempenho proporcionalmente com máquinas adicionais. O *hardware* usado para este *cluster* pode ser qualquer um existente no mercado. Um sistema de nós de computação tão simples quanto ter dois computadores ligados em rede, com um sistema operativo Linux e um sistema de partilha de ficheiros ou tão complexo como ter 1024 nós de alta velocidade e baixa latência da rede.

Figura 3.1: Exemplos de Clusters Beowulf: À Esquerda Cluster Beowulf da NASA com 64 PC's comuns. À direita o exemplo de um Cluster da FutureWare. [18] [14]

Apenas como curiosidade, não só dados científicos que são processados nestes *clusters*, existem inclusive casos de filmes como o ***Shrek*** e o ***Final Fantasy***, que foram renderizados inteiramente utilizando *clusters Beowulf*.

Os sistemas fortemente acoplados têm melhor desempenho e são fisicamente mais pequenos que os sistemas fracamente acoplados, mas como ponto negativo têm um investimento inicial muito maior e podem ter uma depreciação mais rápida, isto é, o tempo que uma determinada máquina leva a perder o seu valor. No caso dos nós de um *cluster* isso não acontece, pois a produtividade do sistema, depende do numero de nós que seja adicionado. Normalmente são usados computadores dados como dispensáveis, embora possam fazer parte do sistema máquinas que sejam estações de trabalho. Neste último caso as máquinas apenas entram no

processamento do *cluster* quando estão inactivas, optimizando assim ao máximo a utilização da estação, trabalho comum e processamento de dados.

3.3 *Clusters* de Computadores

Um *Cluster* de computadores segundo a definição de Andrew Tanenbaum [15] é um conjunto de máquinas independentes que se apresenta ao utilizador como um sistema único e consistente. Em adição a este definição, George Coulouris[22] ainda acrescenta "conjunto de máquinas independentes interligados através de uma rede de computadores equipados com software que permita a partilha de recursos do sistema: hardware, software e dados."

Começando por falar no hardware, um *cluster* é um conjunto de computadores (heterogéneos ou não) conectados através de placas de rede que se destina ao processamento paralelo, ou seja, as máquinas trabalham como uma única máquina, com distribuição de processos e dados entre elas.

O grande objectivo de um *cluster* é possibilitar o uso de computadores ligados em rede para execução de processamento com alto desempenho, permitindo a realização de simulações e processamento avançadas. O processamento paralelo consiste em dividir uma tarefa nas suas partes independentes e na execução de cada uma destas partes em diferentes processadores.

Antes da utilização de *Clusters* de processamento paralelo é necessário que se conheçam os vários tipos de processamento paralelo, algumas das topologias usadas e as bibliotecas usadas para a comunicação entre as máquinas.

3.4 Tipos de Processamento Paralelo

A classificação dos tipos de processamento paralelo depende de vários parâmetros, nomeadamente Topologias, Distribuição de Trabalho, Memória e Hierarquia.

3.4.1 Topologias de Rede/Comunicação

Relativamente às topologias de rede usadas em processamento paralelo temos, entre outras,

- Anel
- Árvore
- Estrela
- Hipercubo
- Malha

Anel

Na topologia em anel, cada posto está directamente ligado a dois outros postos da rede. Os dados circulam num sentido de um posto para outro, cada posto inclui um dispositivo de recepção e transmissão, o que lhe permite receber o sinal e passá-lo ao posto seguinte, no caso da informação não lhe ser destinada.

O primeiro computador da rede vai guardar o *token* que é uma licença para os computadores comunicarem. O computador que desejar efectuar uma comunicação vai requisitar o

token[1] e só depois vai efectuar a transmissão de dados. Desta forma o numero de colisões é reduzido o que permite um maior aproveitamento do meio de comunicação (Figura 3.4.1).

As principais vantagens desta topologia são a necessidade de um pequeno comprimento de cabo e o facto de não serem necessários armários de distribuição de cabos dado que as ligações são efectuadas no próprio nó. Referindo apenas duas desvantagens desta topologia as que se destacam são a falibilidade da rede, basta que um nó deixe de funcionar para que toda a rede falhe; ainda o facto de haver uma maior dificuldade na localização de falhas.

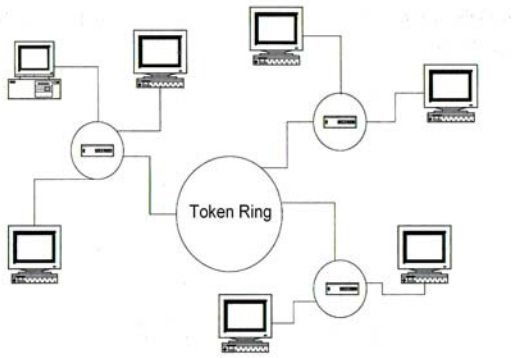

Figura 3.2: Topologia em Anel.[8]

Árvore

A topologia em árvore é essencialmente uma série de nós interconectados. Geralmente existe um nó central onde outros nós com menor importância se conectam. Esta ligação é realizada através de *switches* e das conexões das estações realizadas do mesmo modo da topologia de barramento (Figura 3.4.1).

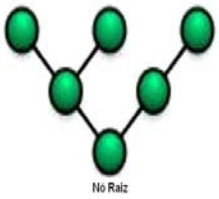

Figura 3.3: Topologia em Árvore.[**?**]

[1]Semelhante à Passagem do Testemunho no Atletismo. Quando um computador recebe o Testemunho pode transmitir a informação

Estrela

No caso da topologia em estrela os postos ligam-se todos a um ponto central, que é um dispositivo que pode ser um *hub* ou um *switch*; em qualquer dos casos esse dispositivo actua como um concentrador (Figura 3.4.1).

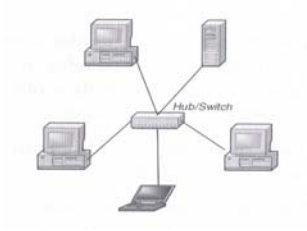

Figura 3.4: Topologia em Estrela.[8]

O concentrador tem como função receber os sinais provenientes dos vários computadores e enviá-los ao computador de destino. Este tipo de topologia tem como principais vantagens, a facilidade de modificação do sistema, já que todos os cabos convergem para um só ponto; fácil detecção e isolamento de falhas, dado que o nó central está ligado directamente a todos os outros. No entanto existem também algumas desvantagens, como é o caso da necessidade de maior comprimento de cabos para efectuar ligações e o facto de todos os nós dependerem apenas do nó central, sendo que se este falha, toda a rede falha.

Hipercubo

A topologia em hipercubo consiste em que o caminho entre os dois nós mais distantes tem no máximo $\boldsymbol{n}$ ligações, onde $\boldsymbol{n}$ é a dimensão do cubo. Desta forma cada nó tem $\boldsymbol{n}$ ligações. Pode ver-se na figura seguinte como é construída esta topologia. Cada ponto corresponde a um processador. No caso de um ponto apenas existe um processador, assim para se construir uma dimensão aplica-se uma fórmula de definição

$$2^n. \tag{3.1}$$

Onde o resultado da fórmula é o número de processadores que existem na rede.

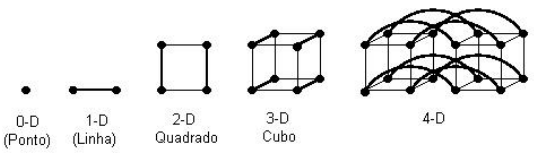

Figura 3.5: Topologia em Hipercubo.[8]

Malha

Na topologia Malha existe uma ligação física directa entre cada um dos nós, isto é, todos comunicam com todos. Embora muito pouco usada em redes locais, uma variante da topologia malha, a malha híbrida, é usada na *Internet* e em algumas *WANs*.

Na figura seguinte podemos verificar a complexidade desta topologia. É um exemplo de quatro nós e já se torna relativamente complexa, mas isso aumenta exponencialmente conforme são acrescentados nós (Figura 3.4.1).

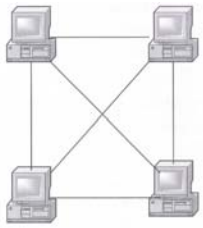

Figura 3.6: Topologia em Malha.[8]

A principal vantagem desta topologia é a elevada tolerância a falhas. Relativamente às desvantagens temos a complexidade de ligações e da rede à medida que aumenta o numero de computadores na rede; por exemplo se uma rede tiver dez nós, existirão cerca de 45 ligações. Como se pode verificar aumenta imenso a complexidade, nunca esquecendo que todos os computadores comunicam entre si.

3.4.2 Modelo de Programação

No que diz respeito a este tópico temos essencialmente dois tipos: Decomposição de Dados e Decomposição de Funções. Apenas para dar uma ideia dos tipos de distribuição e do funcionamento dos seus algoritmos, segue-se uma pequena descrição de cada um deles.

Decomposição de Dados

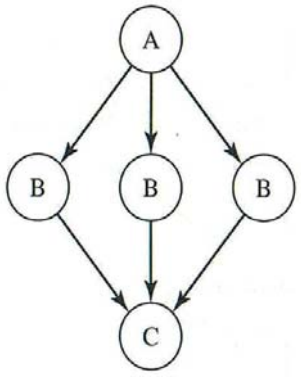

Figura 3.7: Exemplo da Paralelização baseada em Dados. [1]

Este tipo de distribuição é aplicado quando a mesma operação é realizada num conjunto de dados. Ou seja podem-se dividir os dados e processá-los de forma independente. Imaginemos

um determinado processo a que pode ser dividido em dois, b e c. Processando os dados em múltiplas máquinas temos b processado numa máquina e c processado noutra. Assim $a = b + c$

Decomposição de Funções

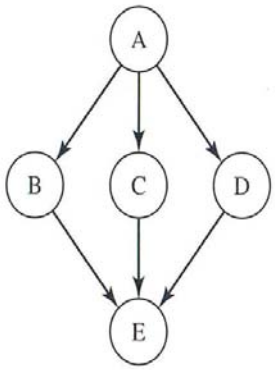

Figura 3.8: Exemplo da Paralelização baseada em Funções. [1]

Um grafo de dados dependente mostra um paralelismo funcional quando existem tarefas independentes aplicadas a operações diferentes, em diferentes elementos dos dados.

3.4.3 Memória

No caso da memória existem essencialmente dois tipos: **memória distribuída** e **memória partilhada**.

Memória Distribuída

Denomina-se Memória Distribuída quando temos memória associada a cada processador. Normalmente usado para um número muito elevado de processadores. Neste caso o tempo de acesso à memória não é uniforme. O acesso aos dados na memória local é mais rápido que o acesso a memórias "remotas". É uma arquitectura denominada NUMA (*non-uniform memory access*).

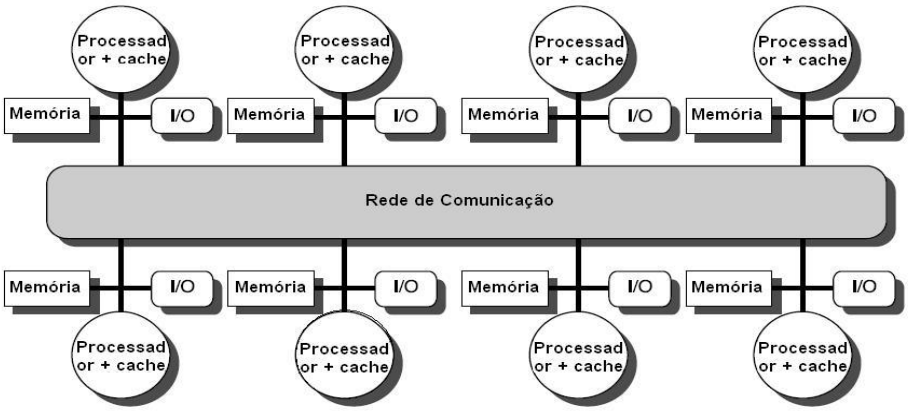

Figura 3.9: Esquema de Memória Distribuída

Memória Partilhada

A memória Partilhada, por sua vez, está ligada a um numero de processadores mais pequeno, sendo possível neste caso, usar um sistema de memória centralizado, através de um barramento comum. Cada processador tem uma memória cache, que sendo suficientemente grande, permite reduzir o numero de acessos aos aos recursos comuns (barramento de dados e memória partilhada). O tempo de acesso à memória neste caso é sempre o mesmo. Temos uma arquitectura denominada UMA (*uniform memory access*).

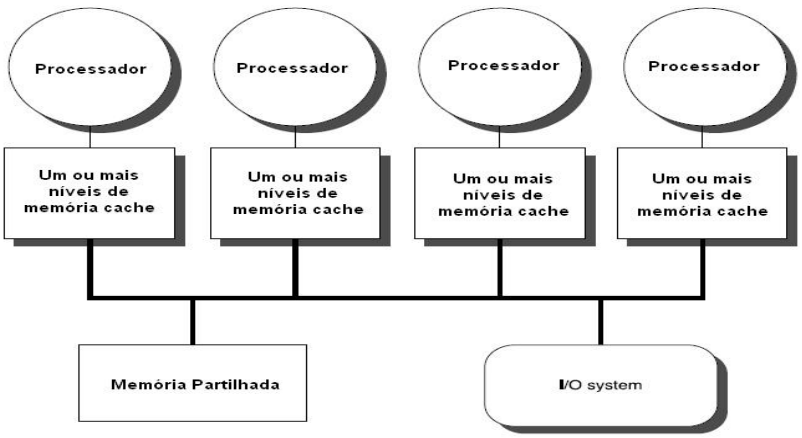

Figura 3.10: Esquema de Memória Partilhada

3.4.4 Hierarquia

Finalmente, falando da hierarquia na Computação paralela, esta pode ser: Master-Slave e Node-Only.

Master-Slave

É um modelo no qual o programa de controlo encarrega o *master node* pelo processo de inicialização, distribuição, escolha e apresentação dos resultados. Algumas vezes, este pode ser também responsável pelo *timing* das funções. Os *slave nodes* realizam toda a computação necessária; o *master* aloca o seu fluxo de trabalho em cada um dos *slave nodes* (estática ou dinamicamente) ou estes realizam a alocação de si próprios.

Node-Only

Num modelo *node-only* são executadas múltiplas instâncias de um mesmo programa, com um processo a ocupar-se de todo o trabalho não ocupacional, ao invés de contribuir para a computação.

Modelos Híbridos

De referir que todas estas características apresentadas no ponto 3.4 podem ser conjugadas dentro dos vários grupos, formando assim modelos híbridos (ex.: anel + estrela) ou entre os vários grupos, conseguindo assim uma enorme variedade de combinações para funcionamento em computação paralela.

3.5 Desempenho e Critérios de Paralelização

Quando um multi-processador opera no máximo do seu desempenho:

- todos os processadores realizam trabalho útil;
- não existem processadores ociosos;
- não são realizadas operações duplicadas.

Nesta situação, os n processadores que compõem o computador paralelo contribuem efectivamente para o desempenho global:

$$\textit{Tempo de execução} = \frac{\textit{Solução sequencial}}{n}$$

Na maioria das aplicações, o desempenho ideal (máximo) raramente é atingido, excepto para algumas aplicações assíncronas.

3.5.1 Factores que contribuem para a diminuição de eficiência

Basicamente são três os factores que contribuem para a queda da eficiência:

1. "*Overhead*"de comunicação entre processadores;
   - depende da taxa de granularidade. (Ver 3.5.2)
2. "*Overhead*"devido ao nível de sincronismo entre tarefas
   - relacionado com a dependência das tarefas alocadas em processadores diferentes, na solução de um problema.

- Leva a desbalanceamento de carga entre processadores.

3. "*Overhead*"devido ao esforço despendido por alguns processadores quando mais de um executa a mesma tarefa.

    - relacionado com a coordenação do processo.

Como lidar com esses factores

Na implementação paralela da solução de um problema, estes factores deverão ser levados em conta, logo, a decomposição do problema em tarefas fracamente acopladas, assim como uma boa coordenação do processo paralelo deverá ser realizado visando minimizar o impacto dos factores (1 - 3).

3.5.2 Granularidade

A taxa de granularidade é dada pela razão:

$TG = P/C$

onde:
P: tempo de processamento local (do processador); C: tempo de comunicação entre processadores.

Paralelismo de alta granularidade: implica em uma TG elevada, onde o tempo de processamento é muito maior do que o tempo gasto para os processos se comunicarem.

Paralelismo de grão fino: TG muito pequena; o "overhead"de comunicação é relativamente elevado com relação ao processamento local.

-> *"Uma boa decomposição deve identificar tarefas com altas taxas TG.*

3.5.3 Lei de *Amdahl*

Os problemas de ineficiência tendem a crescer com o número de processadores; Na prática, observa-se que a eficiência decresce rapidamente, levando à saturação da curva de ganho:

- "aumento de processadores activos não consegue diminuir o tempo de processamento global."

Eventualmente, pode-se ter ganho negativo:

- "A participação de mais processadores causa aumento do *overhead* de comunicação."

Uma estimativa do ganho máximo a ser obtido num algoritmo paralelo pode ser conseguido a partir da Lei de *Amdahl*:

$$G = \frac{T}{Ts + \frac{Tp}{n}} \tag{3.2}$$

onde:

T: Tempo de execução sequencial do algoritmo completo;
Ts: Tempo de execução da porção não paralelizável do programa;
Tp: Tempo de execução da porção paralelizável do programa.

Forma Alternativa da Lei de *Amdahl*

$G = 1\frac{}{fs+\frac{fp}{n}}$

Onde:
fs e fp: fracções não paralelizável e paralelizável, respectivamente, do algoritmo sequencial.

- Quando n cresce, o limite superior de G é 1/fs:
  - "O máximo ganho do algoritmo paralelo estará limitado pela fracção não paralelizável do problema."

Comentários: Existem algoritmos paralelos com poucas operações sequenciais; logo a Lei de *Amdahl* é útil como uma forma de determinar se um algoritmo é um bom candidato para ser paralelizado.

3.6 Medidas de qualidade de algoritmos paralelos

As duas medidas mais utilizadas são:

- Ganho ou *Speed-up*
- Eficiência

3.6.1 *Speed-up* ou Ganho

É definido como:

$S = ts/tp$

Onde:
ts: Tempo de execução da melhor versão sequencial; tp: Tempo de execução do algoritmo paralelo, utilizando p processadores.

3.6.2 Eficiência

É definida como:

*E(%) = (S/p) * 100*

Onde:
S: *speed-up*; p: número de processadores.

3.7 Bibliotecas para o Desenvolvimento de Processamento Paralelo

Outro aspecto importante na paralelização são as bibliotecas existentes para o Desenvolvimento de processamento paralelo. Existem algumas bibliotecas usadas para a comunicação em processamento paralelo, nomeadamente:

- EXPRESS
- LINDA
- MPI
- P4
- PVM

No entanto os pacotes de programação paralela mais usados nos dias de hoje são o PVM e o MPI, pelo que se falará apenas destes dois, com mais ênfase no MPI como sendo o pacote usado no SPW, software usado para a realização desta tese. Ambas as linguagens são baseadas no paradigma de troca de mensagens, o que permite explorar a computação paralela numa grande variedade computadores.

3.7.1 PVM - *Parallel Virtual Machine*

O PVM é um sistema de software iniciado no Verão de 1989 no *Oak Ridge National Laboratory*. Este visa permitir que um grupo de computadores heterogéneos se liguem através de uma rede e passem a ser vistos por um utilizador, com apenas um único computador virtual paralelo.

Assim, os grandes problemas computacionais podem ser resolvidos de modo mais eficaz e custo reduzido, utilizando a potência global e memória de muitos computadores. O PVM é um software com grande capacidade de portabilidade. O PVM permite aos utilizadores explorar o hardware existente no seu próprio computador para resolver problemas muito maiores com o menor custo adicional. O PVM é composto, basicamente, de três partes:

- uma biblioteca de funções (com *Bindings* para C, C++ e FORTRAN) que implementam para o utilizador as directivas de programação da Máquina Virtual (MV);
- um processo *daemon* que irá correr em cada *host* participante da MV;
- uma consola a partir da qual podem ser executadas algumas funções básicas de controle (configuração da MV, gestão de tarefas, etc...).

A implementação do PVM é baseada em processos do Unix. Na verdade, cada tarefa PVM é um processo Unix. Isto explica parcialmente a alta portabilidade do sistema para computadores de arquitecturas tão diferentes. Tarefas são as unidades básicas de execução do PVM. É verdade que, em algumas implementações, as tarefas podem não ser processos. Nestes casos, caberá ao implementador garantir a compatibilidade (tais excepções são mais comuns em sistemas de natureza mais complexa como computadores paralelos). Os programas em PVM podem correr espalhados por uma rede de natureza heterogénea. Mais particularmente, é possível disparar tarefas em computadores em qualquer parte da Internet (desde que o utilizador tenha acesso a esta máquina, obviamente). Esta independência em relação à rede de comunicações que liga os hosts que participam da MV é garantida pelo uso do protocolo TCP/IP para comunicação entre as tarefas através da rede.

Resumindo, o PVM:

- É um modelo de troca de mensagens explicito.
- Tem suporte à heterogeneidade.
- Usa Linguagens: C, C++, FORTRAN.
- Tem um determinado nível de tolerância a falhas.

É composto por duas partes:

1. Um *Daemon* (pvmd3 ou pvmd): residente em todas as máquinas que formam o conjunto da máquina paralela virtual.
2. Uma Biblioteca de rotinas de interface PVM - Devem ser "linkadas"com o programa a ser paralelizado.

O pacote PVM é relativamente pequeno (cerca de 4.5 MB de código fonte em C), e necessita ser instalado apenas uma vez em cada máquina para ser acessível à todos os usuários. Além disso, a instalação não requer privilégios especiais e pode ser efectuada por qualquer utilizador. O PVM pode ser obtido nos seguintes sítios:
- *ftp://netlib2.cs.utk.edu/pvm3/*
- *http://www.netlib.org/pvm3/index.html*

3.7.2 Message Passing Interface (MPI)

A primeira versão standard deste sistema foi desenvolvida por um grupo de investigadores académicos e da indústria em meados de 1994, com o intuito de ser usada numa ampla gama de computadores paralelos. Este standard define a sintaxe e a semântica de um núcleo de rotinas de uma biblioteca de forma a que possam ser usadas numa variedade de programas de troca de mensagens, atingindo assim um elevado nível de portabilidade relativamente aos vários sistemas usados. O MPI é oferecido como parte integrante de alguns sistemas contando com várias implementações dedicadas a redes heterogéneas de *workstations* e multi-processadores simétricos, tanto para Unix como para versões NT do Windows e seus sucessores (ex.: *Server 2003* usado nesta tese).

Essencialmente o MPI é uma plataforma de comunicação, que já vai na sua versão 2.1 (MPI-2.1) que inclui

(MPI-1/2)

- Comunicação Ponto a Ponto.
- Operações colectivas.
- Grupos de Processos.
- Domínios de Comunicação.
- Topologias de Processos.
- Gestão do Ambiente.
- Interface para *profiling*.
- *Bindings* (ligações ou referências) para Fortran e C.

(MPI-2)

- Gestão dinâmica de Processos / Gestão de Processos Dinâmicos
- Suporte para I/O (Input/Output).
- Operações uni-laterais.
- *Bindings* para C++.

Concluindo, tanto o PVM como o MPI têm objectivos semelhantes, um menos comercial que o outro. Verifica-se que a computação paralela tem uma enorme variedade de possibilidades, começando pelas topologias que podem ser usadas, passando pelos critérios de avaliação e ainda pelas bibliotecas de paralelização existentes que podem ser usadas. Desta forma, qualquer utilizador que queira começar no mundo da programação paralela não fica limitado à rigidez de uma só topologia, biblioteca, sistema operativo, ..., mas pode optar por uma diversidade de modelos existentes. Mais à frente nesta tese verificar-se-á que relativamente a alguns modelos apresentados anteriormente são combinados formando uma modelo híbrido. No capitulo seguinte apresenta-se o software usado para processamento sísmico (*Seismic Processing Workshop*) onde será abordado o tipo de paralelização usada. O tipo de Topologias usadas será abordado no Capitulo do *Windows Compute Cluster Server*.

Capítulo 4

Software de Processamento de Dados Sísmicos e Sistema Operativo

4.1 Introdução

Para o processamento dos dados de reflexão sísmica multi-canal, na configuração implementada neste trabalho foram usados como sistema operativo base o *Windows Compute Cluster Server 2003* com *Compute Cluster Pack*.
Neste capítulo serão abordados estes softwares e as suas características.

4.2 SPW - Seismic Processing Workshop

O *software* de processamento de dados sísmicos, é um software de alta *performance* para processamento de dados sísmicos, com uma interface intuitiva e interactiva, desenvolvido pela Parallel Geoscience Corporation. Este foi projectado para ambientes de escritório e/ou trabalho em rede, e conta com suporte de plots sísmicos de alta resolução, montagens e visualizações.
Para esta dissertação, apenas parte do pacote do **SPW** é usada. Assim sendo será referido apenas o **SPW *Flowchart and Executor***.

O SPW *Flowchart* e o Executor trabalham em conjunto para preparar, elaborar e executar os vários passos do processamento sísmico do SPW. O *Flowchart* é um construtor de fluxos flexível e interactivo, que simplifica a entrada de parâmetros em caixas de diálogo e tem ainda folhas de cálculo totalmente editáveis de forma a que se possam alterar as informações do cabeçalho do ficheiro, bem como os dados dos cartões de auxiliares (as informações do cabeçalho dizem respeito ao número de canais de cada registo, número de receptores e número de *shots* do ficheiro SPW).

O *Executor* é um programa *multi-threaded*, *multi-CPU* para executar fluxos construídos no módulo do ***Flowchart***. Combinados, aproveitam-se da tecnologia mais adiantada de um conjunto de *PC's*, oferecendo assim uma solução com uma boa razão (*ratio*) custo-eficiência para processamento de dados sísmicos.

4.2.1 Processamento usado pelo SPW

Divisão de Dados

A divisão de Dados em SPW é feita de forma sequencial. Os registos ou *records* são processados sequencialmente. Cada processador processa um registo de cada vez e estes têm que ser processados de forma seguida. Imagine-se que existem sete núcleos a processar, desta forma são processados sete registos de cada vez e no fim dos sete primeiros registos são processados os sete seguintes e assim por diante até ao fim de todos os registos. A optimização dos tempos de processamento depende muito dos algoritmos criados para cada processo. No SPW existem processos que se podem tornar mais lentos em *cluster* do que em máquina individual. No entanto, existem processos onde se consegue notar uma excelente optimização deste tipo de processamento. Quando forem apresentados alguns resultados da experiência prática verificar-se-á este mesmo ponto.

4.3 *Windows Compute Cluster Server 2003* - WCCS2003 (*Windows Server 2003 + Compute Cluster Pack*)

Neste capítulo será abordado o sistema operativo utilizado para a realização desta dissertação, o *Windows Compute Cluster Server 2003 x64*.

4.3.1 Âmbito

Este Sistema operativo surgiu pela necessidade de ser criado um super computador para cálculo pesado, integrando Computadores Pessoais.
Ao longo de muitos anos a Microsoft tem providenciado o *Clustering* de alto desempenho e com grande tolerância a falhas, mas com um custo muito elevado. Com o *Windows Compute Cluster Server 2003* (a partir de agora referido apenas como *WCCS2003*) esse custo é menorizado.

A computação de alto desempenho (*High Performance Computing - HPC*) está a ser utilizada com servidores de *Clusters* industriais de uma forma crescente.
Estes *Clusters* têm um alcance que suporta, desde uma quantidade pequena de nós (Computadores Individuais), até um número muito elevado. Ligar, configurar, gerir e visualizar estes nós, assim como providenciar o acesso dos utilizadores pode tornar-se muito complexo.

O *WCCS2003* procura responder a essas necessidades, começando por simplificar o desenvolvimento e gestão destes *Clusters* de alto desempenho, assim como reduzir o custo total de utilização. Isto traduz-se numa aplicação de *clusters* computacionais a um nível pessoal e de grupo de trabalho, mais alargada.

O desenvolvimento é integrado com *Active Directory* e *Windows Server 2003* o que faz surgir um desenvolvimento simples dos nós de computação e da gestão central usando o *Microsoft Managment Console (MMC)*. Os computadores num *Cluster de WCCS2003* são *standards* da indústria, sem requisitos de *hardware* e/ou *software* específicos. Um *cluster* pode ser desenvolvido numa gama variada de topologias de rede de forma a satisfazer as mais variadas necessidades.

4.3.2 Descrição do *Compute Cluster Pack* (CCP)

Antes de começar a descrever este sistema, é de extrema importância apresentar as definições e algumas noções que são necessárias para conseguir perceber o desenvolvimento de um *cluster* computacional com este sistema operativo.

Começando por definir *Cluster*, este consiste num *head node* (Computador Principal, Servidor, Raiz) e um ou mais *Compute Nodes* (Nós de Computação, Clientes, Ramos). O *head node* faz a mediação dos acessos aos recursos do *cluster* e actua como computador/nó único para o desenvolvimento, gestão e agendamento de tarefas do *cluster*. Num funcionamento mínimo e apenas para nível de teste ou propostas de desenvolvimento, um *cluster* pode consistir apenas no *head node* que pode também aceitar tarefas, desempenhando também o papel de *compute node*. Assim verifica-se que um *head node* pode realizar processamento paralelo de acordo com o numero de processadores que tem, permitindo assim algum ganho ao nível do desempenho.

Head Node ou Nó Raíz (Servidor)

Este nó fornece uma interface de utilizador e gestão de serviços do *cluster*. A interface do utilizador contém o *Compute Cluster Administrator* (um *snap-in* do *Microsoft Management Console (**MMC**)*), o *Compute Cluster Job Manager* (interface gráfica de Win32) e possui ainda uma linha de comandos (**CLI**). Os serviços de gestão incluem um de tarefas com gestão de tarefas e recursos, nó de gestão e *Remote Instalation Services (**RIS**)*.

Compute Node ou Nó de Computação (Cliente)

Este é um computador configurado como parte do *cluster*, que fornece recursos computacionais para o utilizador do *Cluster*. Os *Compute Nodes* do *WCCS2003* têm que ter um um Sistema Operativo suportado pelo *Compute Cluster*, embora possam existir *clusters* computacionais com diferentes configurações de *Hardware*.

Job Scheduler (Agendador de Tarefas)

Corre no *Head Node* e é responsável pela admissão de tarefas e gestão de filas de espera, alocação de recursos e execução de tarefas. Este comunica com o *Node Manager* que existe em todos os nós.

Managment Infraestructure (Infraestrutura de gestão)

Esta é parte do *Compute Cluster Pack* e permite que o administrador do *cluster* desenvolva e faça a gestão dos *Compute Nodes* e em todos os *Compute Nodes* do *cluster*, fornecendo as *interfaces* administrativas, de utilizador e linha de comandos para a administração do *Cluster*.

Client Utilities (Utilitários de Cliente)

O *Compute node* (Cliente), está restringido em relação às tarefas que lhe são permitidas a nível de consola remota, nomeadamente, adicionar imagens **RIS**. No entanto é preferível administrar o *cluster* a partir do *Head Node*.

Compute Cluster Administrator* e *Job Manager

Estas interfaces são usadas para operações do *Cluster*, submissão de tarefas e gestão. Podem ser usadas para configurar o *Cluster* e monitorizar a sua actividade e estado. O *Job Manager* é usado para criação, submissão e monitorização de tarefas.

***Comand Line Interface* (CLI)**

Serve para gestão de nós e tarefas. Nesta é também criado um *script* de operações, podendo também, ser usada pelos Administradores.

4.3.3 *Active Directory Domain*

Como foi falado anteriormente neste capítulo, é indispensável que o *Head Node* pertença a um *Active Directory Domain*.
Tal importância deve-se ao facto de o *Active Directory* (AD) ser um software da Microsoft usado em ambientes Windows. O AD é uma centralização de dados que armazena informações sobre a recolha de todos os recursos disponíveis no domínio do *Windows Server 2003*. É uma representação hierárquica de todos os objectos e respectivos atributos disponíveis na rede. Permite aos administradores gerir os recursos da rede, ou seja, computadores, utilizadores, impressoras, pastas partilhadas, etc., de uma forma fácil.
Numa rede grande e complexa, o serviço AD fornece um ponto único de gestão para os administradores, colocando todos os recursos de rede num único lugar. Ele permite que os administradores deleguem funções administrativas, bem como facilita a procura rápida de recursos de rede. É facilmente escalável, ou seja, os administradores podem adicionar uma grande quantidade de recursos sem ter trabalho administrativo adicional. É realizado através da separação do directório de dados, distribuindo-o em outros domínios, cria confiança e relacionamento, assim, fornecer aos utilizadores os benefícios da descentralização, e, ao mesmo tempo, mantendo a administração centralizada.
A estrutura lógica representada pelo AD é composta por florestas, árvores, domínios, unidades organizacionais, bem como objectos individuais. Segue-se a descrição de todas as lógicas dos componentes da estrutura do AD:

Árvore: é uma estrutura hierárquica de múltiplos domínios organizados do *Active Directory* (**AD**). Consiste num domínio raiz e vários domínios subordinados. O primeiro domínio criado numa árvore torna-se o seu filho, o domínio raiz o seu pai. Vários domínios árvores podem ser incluídos numa floresta.

Nas empresas de maior dimensão principalmente sendo distribuídas geograficamente, o sistema será formado por um conjunto de domínios hierarquizados em árvore a partir de um domínio de topo.

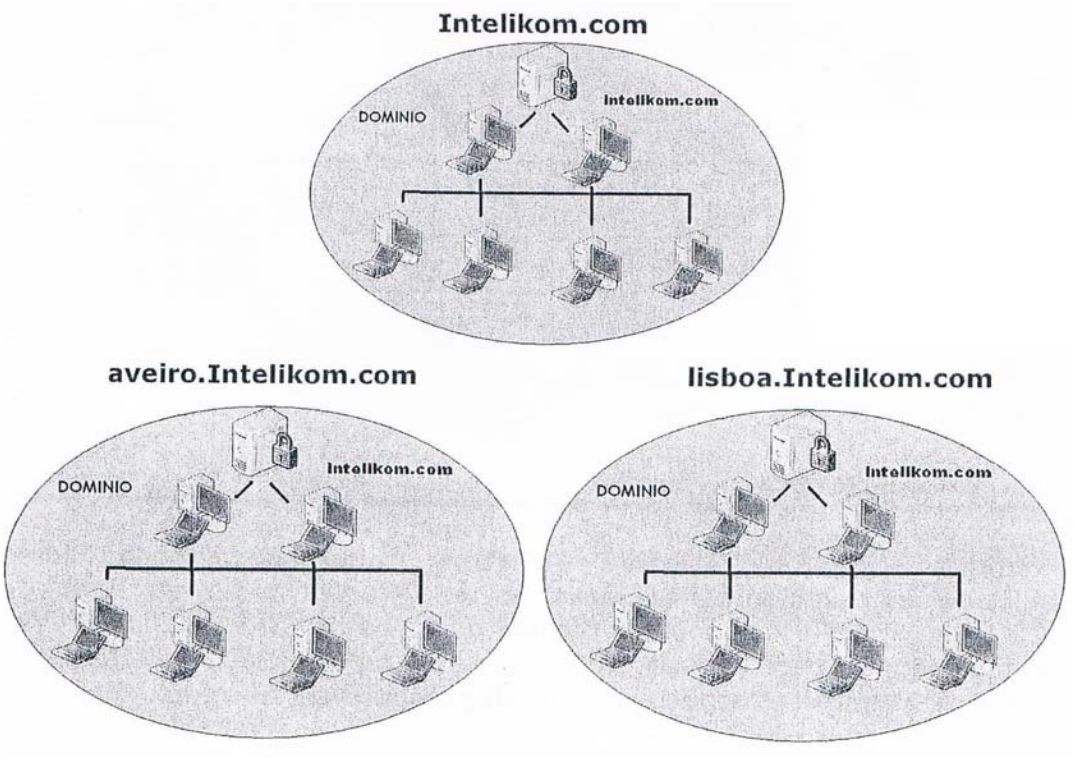

Figura 4.1: Rede de múltiplos domínios, formando uma árvore a partir do domínio Intelikom.

No caso mostrado na Figura4.1 o domínio Intelikom.com seria o domínio de topo, pai de todos os outros, estabelecendo-se assim um modelo hierarquizado em árvore. Cada domínio é uma entidade administrativa, mas existe a possibilidade de gerir (com permissões) o domínio de topo. Depois de instalados os controladores de domínio de cada um deles, existirá uma árvore que partilha um AD disponível para todas as máquinas da árvore. O primeiro *Domain Controlor* (DC) do domínio topo tem de ser o primeiro a ser instalado para que os restantes possam aderir à árvore, pelo que é essencial saber quanto antes o nome do domínio a usar. A árvore poderá ser mais complexa, ou seja, os subdomínios apresentados na figura 4.1 podem também eles conter outros domínios como: **ocidental.aveiro.intelikon.com** e **oriental.aveiro.intelikon.com**. Dentro da árvore, todos os computadores que correm com o *Windows Server 2003* terão acesso ao AD completo, desde que pelo menos um controlador de domínio de cada domínio esteja disponível. Caso contrário, apenas uma porção do directório poderá ser usada.

Objectos: O *Active Directory* armazena todos os recursos da rede, sob a forma dos objectos numa estrutura hierárquica tornando-os facilmente acessíveis e fáceis de gerir.

Unidade organizacional (UO): É uma estrutura que permite a criação de limites administrativos de um domínio, delegando tarefas administrativas distintas para os administradores sobre o domínio. Os administradores podem criar várias unidades organizacionais na rede.

Controlador de domínio: Os DC (*Domain Controllers* - Controladores de Domínio) são os servidores que mantém o directório do sistema, ou seja, a base de dados onde estão definidas as contas dos utilizadores do sistema, os grupos de utilizadores, bem como qualquer outro tipo de informação que o administrador queira armazenar. Estes DC são bastante importantes para o sistema, porque são aqueles que procedem à autenticação dos utilizadores na rede,

ou seja, é aquele que num ambiente de domínio do Windows, o computador que executa o *Active Directory* gere o acesso do utilizador a uma rede, o que inclui o *logon*, a autenticação e o acesso ao directório e aos recursos partilhados. Um DC pode ser demovido do papel bem como ser transferido para outra rede ou domínio. Um controlador de domínio localmente resolve consultas de informações sobre objectos no seu domínio. Um domínio pode ter vários controladores de domínio. Cada controlador de domínio num domínio segue o modelo Multi-maestro por ter um domínio completo réplica da partição do directório. Neste modelo, cada controlador de domínio possui uma cópia original do seu directório partição. Os DC de um domínio formado por servidores trocam entre si as actualizações do directório, que pode ser feita por qualquer um deles. Os administradores podem usar qualquer um dos controladores de domínio do *Active Directory* para modificar a base de dados. As mudanças realizadas pelos administradores são automaticamente replicadas para outros controladores no domínio.

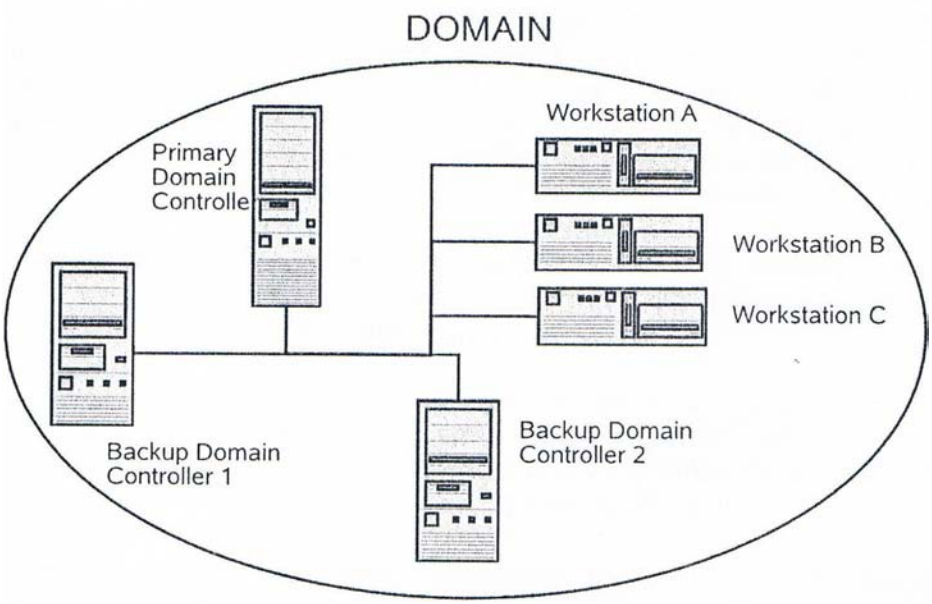

Figura 4.2: Esquema de um domínio com Domain Controllers de Backup.

Floresta: É o limite exterior da estrutura do *Active Directory*. É um grupo de várias árvores. É criado quando o primeiro computador baseado no *Active Directory* está instalado numa rede. Na realidade, quando se instala o primeiro Controlador de Domínio, está-se a criar uma floresta composta por uma árvore, composta por um domínio, por sua vez composto apenas por um controlador de domínio. Existe pelo menos uma floresta em uma rede. O primeiro domínio numa floresta é chamado de domínio raiz. Os administradores podem criar múltiplas florestas e, em seguida, criar confiança nas relações entre os domínios específicos dessas florestas, dependendo das necessidades organizacionais.

Resumo

O **DNS** é um dos serviços mais importantes para o bom funcionamento da Internet. Uma das suas utilizações mais correntes é a tradução de nomes de máquinas nos seus endereços IP e vice-versa. Todos os computadores ligados directamente à Internet têm de ter um endereço IP válido e único nesta rede, ao qual deverá sempre corresponder um nome, atribuído segundo as

regras do **DNS**. Por exemplo, o Servidor de WWW do IPB tem o endereço IP 193.136.195.220, ao qual corresponde o nome *www.ipb.pt*. O DNS é um serviço crítico cuja disponibilidade deve ser mantida elevada, pelo que é conveniente a configuração de secundários, que assegurem o bom funcionamento do serviço mesmo em caso de falha do primário **DNS**. A solução a todos estes problemas é o domínio da rede. Um controlador do domínio tem informações relativas a quem são concedidas as entradas, o que é registado sobre as mesmas, e o que é permitida a cada pessoa fazer na rede. Quando se faz um registo num domínio com um PC, o controlador do domínio verifica as credenciais e permite ou não a entrada. A maioria de redes do domínio tem pelo menos dois controladores do domínio com informação idêntica, assim se um deles deixar de funcionar, o outro pode assumir o controlo. Esta redundância é critica para a segurança de uma rede, pois se o controlador não estiver operacional, toda a rede fica inoperante.

4.3.4 Topologias de Rede em WCCS2003

No *Windows Compute Cluster 2003* existem alguma topologias de rede, como foram já acima referidas que se descrevem de seguida.

Topologia 1:

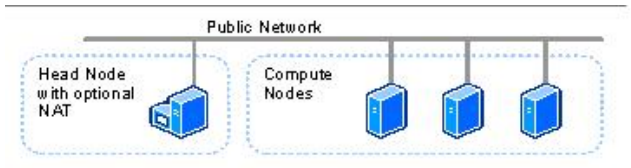

Figura 4.3: Esquema da primeira topologia usada em CCP. *All nodes on public Network.*

Nesta topologia todos os nós pertencem apenas à rede pública, e todo o tráfego é enviado pela rede pública. Esta configuração não suporta o desenvolvimento automático de *compute nodes* através do *Remote Installation Service* (**RIS**).

Topologia 2:

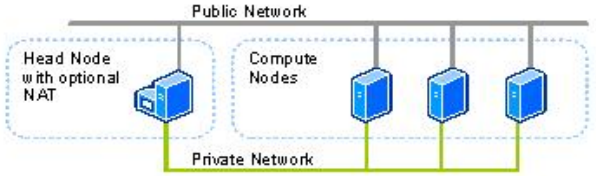

Figura 4.4: Esquema da segunda topologia usada em CCP. *All nodes on public and private Network.*

Nesta configuração, todos os nós estão ligados tanto à rede pública como a privada. O

tráfego interno do *Cluster*, assim como o tráfego MPI é enviado pela rede privada.

Topologia 3:

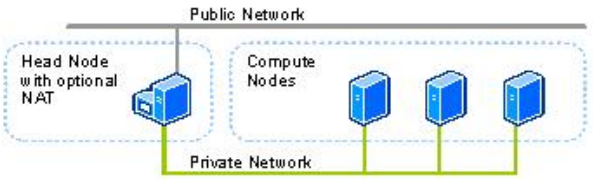

Figura 4.5: Esquema da segunda topologia usada em CCP. *Compute Nodes isolated on private Network.*

Nesta configuração apenas o *Head Node* está ligado à rede pública, enquanto que os *Compute Nodes*, acedem à rede pública através do NAT no *Head Node*. Os tráfegos MPI e do *cluster* são enviados através da rede privada.

Topologia 4:

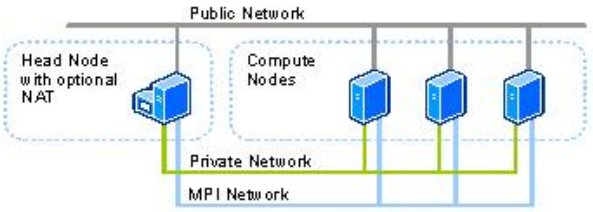

Figura 4.6: Esquema da segunda topologia usada em CCP. *All nodes on Public, Private and MPI Networks.*

Nesta topologia os nós estão ligados às redes pública, privada e MPI. O tráfego interno do *Cluster* é enviado através da rede privada, o tráfego MPI é enviado através da rede MPI.

Topologia 5:

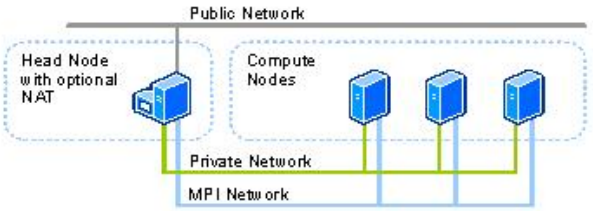

Figura 4.7: Esquema da segunda topologia usada em CCP. *Compute Nodes isolated on Private and MPI Networks.*

Nesta configuração apenas o head node está ligado à rede pública, enquanto que os *Compute Nodes*, acedem à rede pública através do NAT no *Head Node*. O tráfego interno do *Cluster* é enviado através da rede privada, o tráfego **MPI** é enviado através da rede **MPI**.

4.3.5 Outras considerações do WCCS2003

O **WCCS2003** permite que existam máquinas externas ao *Cluster* que necessitem de submeter trabalhos no mesmo, usando assim os seus recursos. Para que isto seja possível, o **PC** que pretende usar os recursos do *Cluster* tem que ter instalado em primeiro lugar, um Sistema Operativo compatível e ainda o **CCP**. De referir que este *Pack* corre também em máquinas de arquitectura x86, com sistema operativo *Windows XP 32bit Service Pack 2* ou superior. Nestas máquinas o utilizador tem apenas as ferramentas que permitem submeter trabalhos e ver os resultados.

É necessário ainda que exista um disco de grande capacidade no *Head Node*, pois é neste que são armazenados os dados processados de todos os *Compute Nodes*. Este disco será um disco de rede partilhado por todos os **PCs**. *(Shared Disc)*

Com um número pequeno de computadores no *Cluster*, o *Head Node* funciona também como *Compute Node*. Com um número superior de máquinas (superior a 10) é preferível usar o *Head Node* apenas como *Head Node*, para optimizar o funcionamento do *Cluster*, caso em que os *Compute Nodes* tratam apenas do processamento e o *Head Node* da gestão do *Cluster*.

4.4 Metodologias e Opções de Configuração

Depois de uma breve introdução ao software, topologias e processos de análise utilizadas nesta tese apresentam-se e discutem-se as opções tomadas relativas a hardware e software que foram configuradas para o melhor funcionamento do *Cluster*.

4.4.1 Hardware

Todos os testes foram realizados em máquinas com as seguintes características:

Intel(R) Core(TM)2 Quad CPU
Q6600 @ 2.40GHz
2.40 GHz, 8,00 GB de RAM

num total de 4 máquinas contendo um total de 16 processadores. Estas máquinas estão ligadas de acordo com a **Topologia 2** do ***WCCS2003***[1] através de um *switch* de rede *Gigabit*. Como se viu anteriormente esta topologia apresenta um esquema no qual cada máquina está ligada a duas redes, uma privada e outra pública. A rede pública está ligada através de uma interface de **100Mbit** e a rede privada que vai conter todo o fluxo de comunicação do *cluster*, a *Gigabit*. Os cabos de rede usados são de **Categoria 5E** de forma a suportarem esta velocidade. O *switch* usado é um **Asus GigaX 1105N**.

[1] Secção 4.3.4 deste trabalho

4.4.2 Software

O Sistema Operativo usado foi o **Windows Server 2003** com o *Compute Cluster Pack*. No inicio deste trabalho, havia a possibilidade de escolha relativamente ao Sistema Operativo a ser implementado. Esta escolha estava no uso do *Server 2003* ou da nova versão lançada pela *Microsoft* também para este tipo de sistemas, o *Windows Server 2008* com o *High Performance Computing* (HPC). Optou-se pelo *Windows Server 2003*, dado que era um sistema mais estável e testado pelos criadores do *Seismic Processing Workshop*. O HPC estava ainda em fase de testes e não foi considerado como a melhor opção para montar esta tese. Por outro lado o *Software* SPW ainda não estava preparado para esta nova versão.
Em relação à topologia usada é uma topologia mista com base Estrela, que faz a comunicação entre o cluster, ou seja o tráfego MPI (troca de mensagens de controlo do protocolo) e as transferências de dados. Optou-se por esta topologia pois por um lado liberta a rede pública do tráfego interno, fazendo com que este seja mais rapidamente transmitido entre máquinas intra *cluster* e por outro lado uma vez que as máquinas já trazem por defeito placas de rede *Gigabit* optimiza-se o funcionamento do *cluster*, minimizando o custo dos dispositivos de rede. Assim a **topologia 2** foi a opção que se achou satisfazer melhor as necessidades requeridas.

Capítulo 5

Apresentação e Análise de Resultados

5.1 Introdução

Depois de explicadas as opções tomadas em relação ao software usado ao longo desta Tese e os processos de análise de algoritmos, passar-se-á a falar dos resultados obtidos com base nas experiências efectuadas. Relativamente aos testes efectuados, como se pode compreender, não foram testados a passos de processamento que o SPW fornece dado o seu extenso número. Foram apenas testados alguns processos mais comummente usados, de forma que sejam fornecidos resultados que melhorem o desempenho temporal dos fluxos usados no laboratório.

5.2 Apresentação de Resultados

Começou-se por testar os fluxos de processamento em modo sequencial, dado que os critérios de avaliação de desempenho em computação paralela (nomeadamente ganho, ou speedup) se baseiam na comparação com esses resultados. Foram usados dados de reflexão sísmica obtidos em três linhas, a saber:

- TGS Nopec704 a norte da costa Africana 8,28 GB (8.479,466 MB)
- t74b na Planície Abissal do Tejo 496 MB
- TGS707 no Golfo de Cadiz 16GB (15.824,539 MB).

Como se pode verificar pelo espaço ocupado em memória, duas delas são comparativamente muito pesadas a nível computacional. Foram aplicados os seguintes processos:

- Geometry Definition;
- Linear Moveout;
- Normal Moveout;
- Radon Demultiple;

- Radon Transform;
- Spherical Divergence Correction;
- Horizontal Median Filter.

A Definição de Geometria (*Geometry Definition*) tem por objectivo ajustar todos os parâmetros de forma a mostrar os locais na posição correcta, corrigir as informações relativas ao posicionamento da fonte e dos receptores, dar informações sobre a distância da fonte a cada receptor, *offset* da fonte. Como mostra na figura 2.1.

O *offset* da fonte diz respeito à real localização da embarcação e à posição onde está colocada a fonte. A embarcação tem um dispositivo de GPS que marca a sua localização, e é através da localização da mesma que são feitas as medições. Desta forma para que a reconstrução da estrutura que se está a estudar esteja bem localizada é preciso fazer os respectivos ajustes de *offset*. Quando são capturados dados tantos os receptores como a fonte são dados como estando à superfície da água, o que na realidade pode não acontecer. Assim têm de ser feitas também estas correcções.

Relativamente ao *Linear Moveout* e ao *Normal Moveout(**NMO**)*, estes são usados para eliminar o ruído da linha e as partes que não interessam no estudo. Para este caso estão a eliminar-se os sinais de incidência directa.

As transformações *Radon* (*Radon Transform* e *Radon Demultiple*) são filtros que se podem aplicar às linhas. O *radon* clássico consiste em transformar um somatório em linha recta por uma série de parâmetros em cada valor de intercepção. [12]

A transformação de Radon é o integral de uma propriedade física (ex.: velocidade, atenuação, densidade) de um objecto ao longo de uma determinada linha ou caminho de energia. É usada em tomografia e no domínio p-t. E é também denominada por Slant Stack. O *Radon Transform* pode então ser usado para calcular o *Radon* linear, parabólico ou hiperbólico, do domínio espaço-tempo para o domínio linear, parabólico ou hiperbólico. O outro realiza um *radon demultiple* parabólico, no qual se podem especificar, o tipo de transformação, a gama de parâmetros de raios na saída da transformação, o comprimento espacial e temporal da taper usada para gerar a transformação.

Finalmente o passo *Horizontal Median* permite que seja aplicado um filtro de mediana horizontal ao longo dos traços dos dados de uma *gather* ou *stack*. Este processo é muito útil no processamento de dados VSP para a separação de eventos com sentidos descendentes e ascendentes.

Os resultados obtidos com cada um dos processos são apresentados nas tabelas 5.1 e 5.2:

Tabela 5.1: Resultados de Processamento em Modo Série

	Geometry Definition	**Linear Moveout**	**Normal Moveout**	**Radon Demultiple**	**Radon Transform**	**Spherical Divergence Correction**	**Horizontal Median Filter**
Modo Série	0:33:16	0:28:58	0:23:50	1:10:57	0:59:12	0:30:06	0:21:49

Tabela 5.2: Resultados de Processamento em Modo Paralelo

	Geometry Definition	**Linear Moveout**	**Normal Moveout**	**Radon Demultiple**	**Radon Transform**	**Spherical Divergence Correction**	**Horizontal Median Filter**
1 PC	0:13:02	0:14:06	0:12:07	0:25:07	0:20:23	0:20:33	0:09:20
2 PC	**01:15:32**	**0:28:54**	**0:32:56**	**0:13:17**	**0:09:39**	**0:31:11**	**0:11:32**
3 PC	1:26:46	0:31:42	2:50:44	0:08:04	0:06:28	0:51:20	0:16:22
4 PC	**3:06:52**	**0:51:10**		**0:06:25**	**0:04:51**		**0:59:46**

Como podemos verificar, ao contrário dos restantes processos, as transformações de *Radon* apresentam melhorias muito significativas de desempenho no funcionamento em *cluster*. Passaremos por isso a analisar em mais detalhe os resultados de eficiência e *speedup* para estes processos. Considerando processamento paralelo com n processadores em circunstâncias ideais, ou seja, com todos os processadores a realizar trabalho útil no máximo do seu desempenho, teremos:

$$\mathrm{t}_{execução} = \frac{t_{sequencial}}{n} \quad (5.1)$$

A tabela 5.3 apresenta os resultados da aplicação desta equação ao caso da Radon Transform para diferentes números de processadores, tendo em conta que, como se pode ver na Tabela 5.1, o tempo de cálculo em modo sequencial (uma única máquina) para esse processo é 00:59:12.

Os testes práticos mostraram resultados muito próximos destes valores teóricos. As diferenças devem-se, como ficou explicado no capítulo 3, ao facto de os processadores não trabalharem no máximo da sua capacidade, na medida em que gastam recursos computacionais para gestão de comunicação entre processos. Aplicando os critérios de avaliação discutidos anteriormente, obtiveram-se os resultados da tabela 5.4.

Tabela 5.3: Resultados de Ganho e Eficiência para a Radon Transform

Tempo Serie	Nº Processadores	Tempo Execução Paralelo (s)	Speedup	Eficiência (percentagem)
	4	1224	2,8	72,02
	8	563	6,3	78,86
	12	389	9,1	76,10
	16	291	12,2	76,29

Pode-se verificar que a paralelização deste tipo de processo alcança excelentes valores de eficiência. A eficiência atinge o pico com 8 processadores, e começa a decrescer a partir desse ponto. Constata-se assim que, se o número de processadores aumentar, o *speedup* não aumentará na mesma razão. O gráfico da Figura 5.1 mostra a evolução dos tempos de processamento em função do aumento do número de processadores.

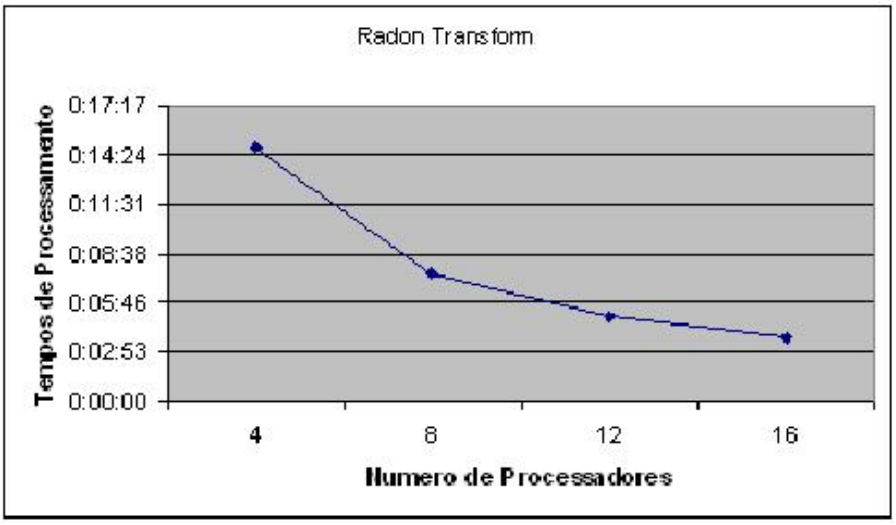

Figura 5.1: Gráfico do Tempo de Processamento vs Nº de Processadores.

Para o processo *Radon Demultiple* realizou-se um estudo análogo. Os resultados em termos de tempos de processamento teóricos e práticos são apresentados na tabela 5.5:

Tabela 5.4: Tempos de Processamento para a *Radon Demultiple*.

Nº de CPU's	Teórico	Prático
Serie		1:10:57
4	0:17:44	0:25:07
8	0:08:52	0:13:17
12	0:05:55	0:08:04
16	0:04:26	0:06:25

As ligeiras diferenças justificam-se da mesma forma: existem processos de troca de mensagens do *cluster* que têm que ser geridas. A análise de desempenho resulta na Tabela 5.6:

Como se pode verificar, tanto o speedup como a eficiência deste processo apresentam excelentes valores, tal como a Radon Transform. Obtivemos de novo uma curva de tempos de processamento em modo paralelo em função do número de processadores, que ilustra esta conclusão.

Tabela 5.5: Resultados de *Speedup* e Eficiência para a *Radon Demultiple*

Tempo Serie	Nº Processadores	Tempo Execução Paralelo (s)	Speedup	Eficiência (percent)
	4	1507	2.7	70.62
	8	797	5,3	66,76
	12	484	8,8	73,29
	16	385	11,1	69,11

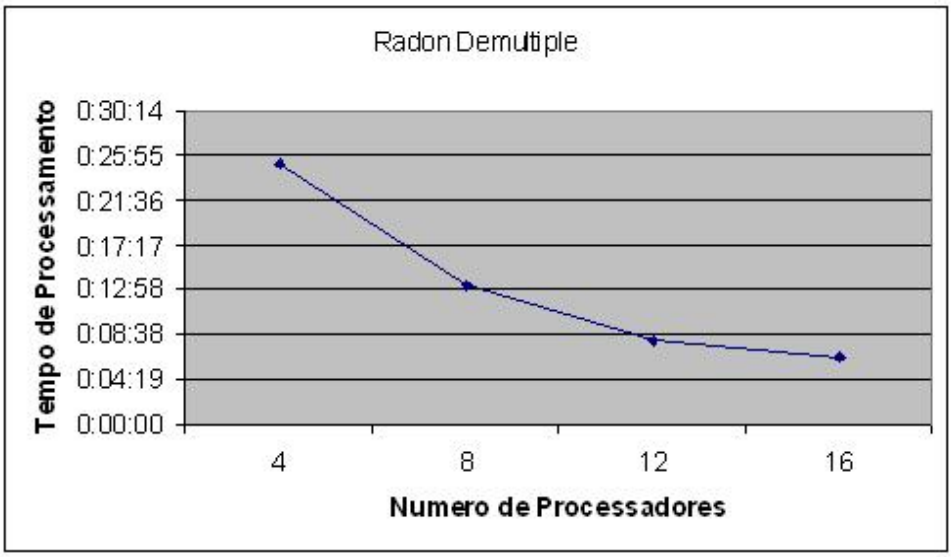

Figura 5.2: Gráfico do Tempo de Processamento vs Nº de Processadores.

Analisando estes dois casos, que se encontram entre os mais pesados computacionalmente e mais importantes no processamento, pode dizer-se que a utilização do cluster oferece grande vantagem. Por exemplo, se um processo demorar uma semana (24h x 5 dias = 120 horas) em modo sequencial, com o uso do cluster passa a ter uma optimização média de 75.4 por cento, ou seja, passa a demorar 29:26 horas (cerca de 1 dia e 4,5 horas).

Os processos que não apresentam melhoria em processamento paralelo são processos muito intensivos em comunicação como se ilustra na figura 5.3. Trata-se assim de problemas de excessivo overhead de comunicação.

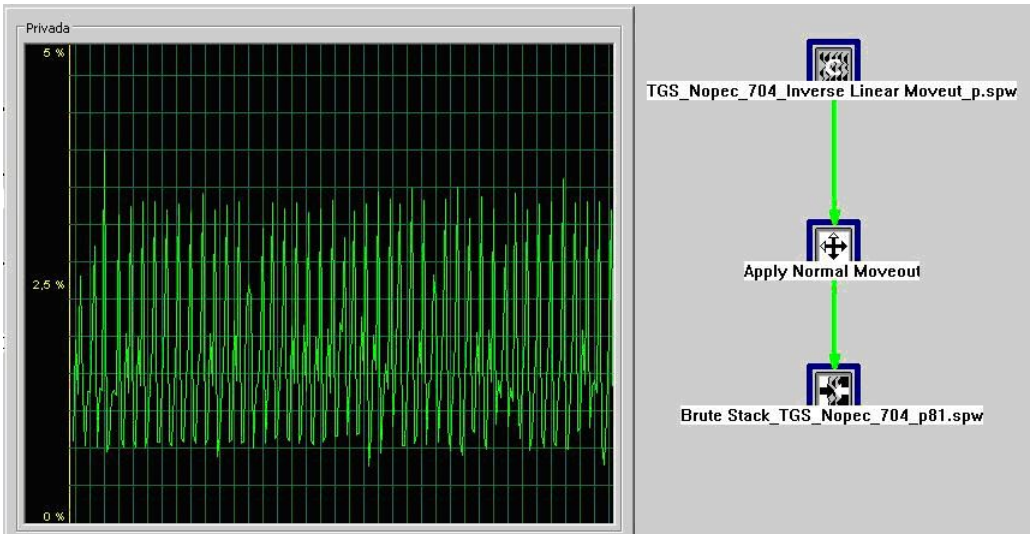

Figura 5.3: Utilização de Rede.

Conclui-se que os algoritmos destes processos não estão optimizados para funcionamento em paralelo. Com este teste verificou-se também o mau funcionamento a nível de gestão dos processadores por parte do Sistema Operativo, já que apenas uma pequena quantidade dos recursos dos processadores é usada (na ordem dos 15 a 20 por cento). Quando paralelizamos usando os quatro processadores da mesma máquina, observamos que essa percentagem sobe drasticamente, para valores a rondar os 90 por cento. Não surpreende por isso a forte diminuição do tempo de processamento, dado que a capacidade total de processamento passa a ser muito melhor aproveitada. Conclui-se por isso que mesmo quando se usa apenas uma máquina, existe grande vantagem em usar o SPW em modo paralelo.

Capítulo 6

Conclusões

Com este trabalho pode concluir-se que a utilização de um *cluster* computacional é uma excelente aposta no processamento de dados de sísmica de reflexão. Observaram-se grandes melhorias a nível de tempo de processamento em processos de crítica importância e o custo de implementação é reduzidíssimo quando comparado com a aquisição de um multi-processador. Salienta-se que a optimização do tempo de processamento dos dados é muito dependente do algoritmo de paralelização usado, da dimensão dos dados e do tipo de operações. Os resultados obtidos em processamento paralelo no *cluster* podem inclusivamente, em certos casos, ser piores do que em processamento sequencial, numa única máquina, como comprovámos em alguns testes. Com efeito, paralelizar um algoritmo que não se preste a funcionar nesse modo contribui para uma queda de eficiência devida aos factores discutidos na secção 3.5.1 desta tese. Porém, quando são usados algoritmos preparados para este tipo de funcionamento, um *cluster* deste tipo é extremamente eficiente, como podemos verificar através de alguns dos resultados do capitulo 5. Um *cluster* oferece também vantagem pela sua flexibilidade: o investimento nas estações de trabalho dos colaboradores, permite, por um lado, aumentar a sua produtividade em tarefas não passíveis de paralelização e, por outro, mobilizar uma grande capacidade de processamento para tarefas especialmente pesadas em horários convenientes (ex: à noite). Esta vantagem é especialmente valiosa num laboratório de investigação.

6.1 Trabalho Futuro

Uma continuação para o trabalho realizado nesta tese será uma migração para o Windows HPC logo que este se mostre com estabilidade suficiente e seja recomendado o seu uso. Com esta migração testar-se-iam outros processos que não se testaram nesta tese assim poder-se-iam tirar conclusões sobre a paralelização de um maior número de processos. Outro possível passo será tentar ligar o *cluster* a máquinas externas ao laboratório de forma a aproveitar a capacidade de processamento de outra máquinas do Campus Académico. Desta forma teríamos um grande aproveitamento da capacidade de computação existente no Campus, permitindo que se realizasse processamento mais pesado e mais exigente computacionalmente. Permitindo, não só, serem processados dados de Geofísica mas também de outras áreas. Por exemplo ao nível de processamento de sinal áudio-acústico.

6.2 Comentário Final

Como comentário final, parece-nos importante prosseguir a aposta em desenvolver teses de mestrado dos cursos de Engenharia do DETI (nomeadamente o de Computadores e Telemática) em ambientes exteriores ao Departamento. Criam-se assim oportunidades excelentes de abrir horizontes e abordar áreas variadas, contribuindo para um enriquecimento curricular e pessoal dos alunos.

Apêndice A

Configurações do *Cluster* em WCCS2003

A.1 Desenvolvimento de um *Cluster* Computacional

Existem alguns requisitos para configurar um *Cluster* Computacional. Em primeiro lugar, deve-se verificar se os computadores que se pretende usar preenchem os requisitos de *Software* e *Hardware*. Depois disto é necessário que se certifique que a topologia escolhida vai de encontro às reais necessidades dos Utilizadores.

Para desenvolver um *cluster* computacional em *Windows* começa-se por definir qual será o *Head Node*. Depois de configurado nessa máquina o *Windows Server 2003* prossegue-se com a configuração do *Compute Cluster Server* (**CCS**).

De fazer notar que as configurações do *WCCS2003* no Servidor devem conter na sua instalação *Active Directory*, DNS e DHCP.

Depois de definido o *Head Node* têm que ser seguidos os seguintes passos:

A.2 Configuração do Head Node

- Instalar o *Compute Cluster Pack* (**CCP**).
- Adicionar o *Head Node* a um *Active Directory Domain*, ou fazer do *Head Node* um Controlador de Dominio para o *Cluster*.
- Configurar as NICs (*Network Interface Cards*).
- Ligar o *Windows Cluster Manager*
  - Definição do *Head Node*
    - *Create New Cluster*
    - *Compute as Head Node*

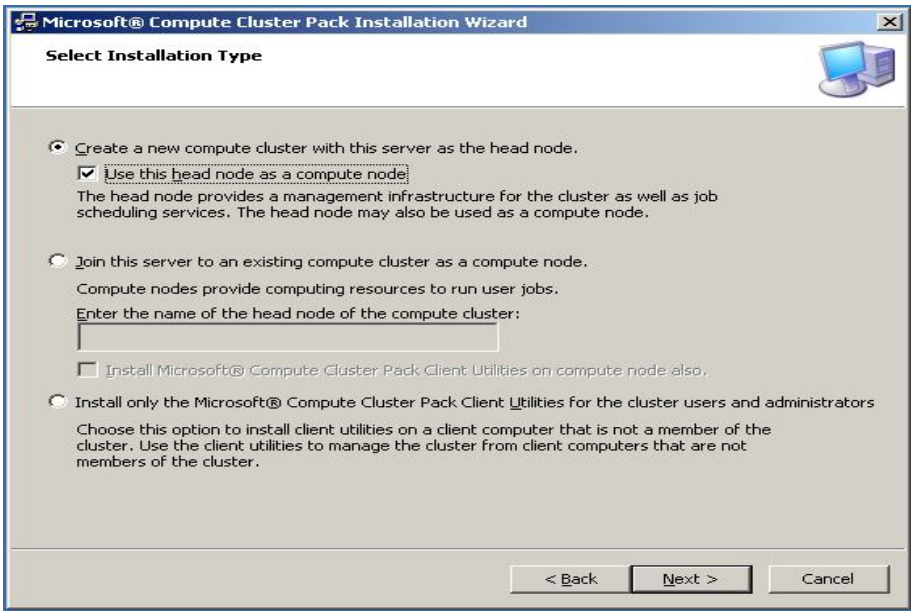

Figura A.1: Selecção de Head Node ou Compute Node.

∗ Instalar todos os serviços não instalados, que são requeridos

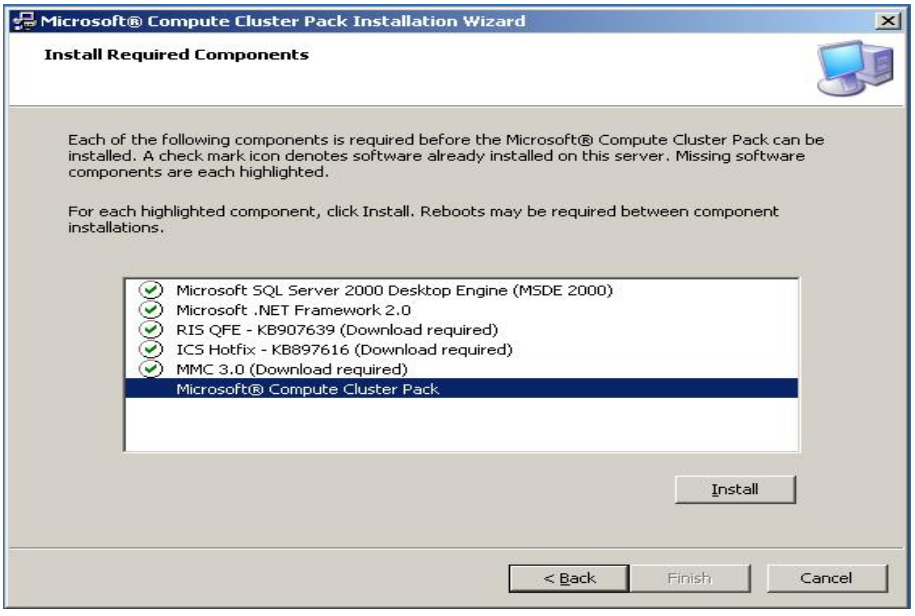

Figura A.2: Lista de Serviços a Instalar.

- Depois de configurado o *Head Node* será necessário configurar os vários pontos da *To do List*.

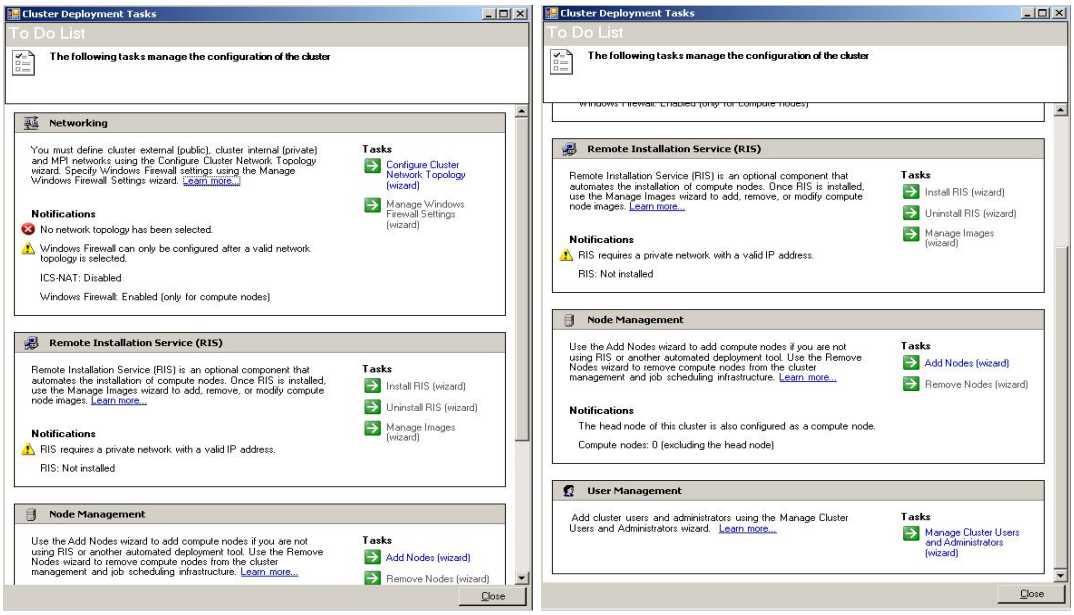

Figura A.3: Lista de Tarefas de Configuração (*To Do List*).

1. Configurar a Topologia pretendida segundo um dos 5 cenários possíveis. (De referir que esta pode ser alterada a qualquer altura, tendo em conta as respectivas alterações a nível de *hardware*.)

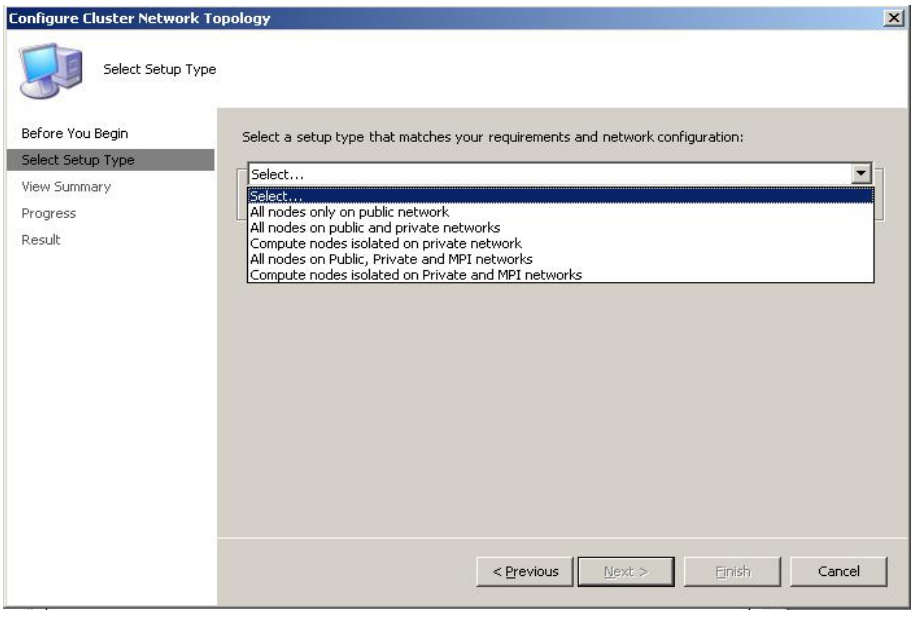

Figura A.4: Lista de Topologias de Rede.

Neste caso concreto foi escolhida a segunda opção, *All nodes on public and private network*.

2. Seleccionar LANS para cada Rede. A que liga à *Private Network* e a que liga à *Public Network*.

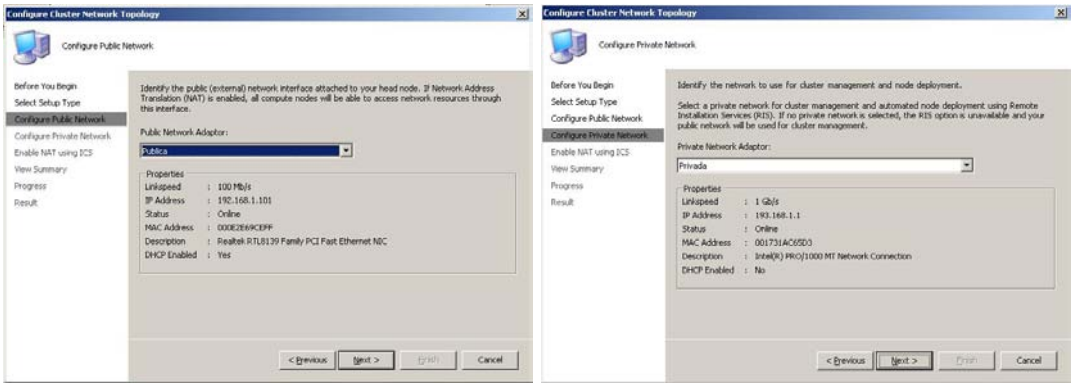

Figura A.5: Selecção das Placas de Rede para as redes pública e privada (*NIC's*).

3. Ligar à Internet.

Além das redes privadas está presente em alguns cenários o **MPI**, cuja função está relacionada com a passagem de mensagens entre o *Head Node* e os vários *Compute Nodes*.

A próxima configuração tem a ver com o **RIS** (Serviços de Instalação Remota). É aconselhável que, a não ser que a rede do campus o exija, este esteja desactivado.

Para finalizar as configurações do *Head Node*, serão adicionados os *Compute Nodes*. Numa fase inicial estes ainda não estão configurados, pelo que têm que ser configurados antes de serem adicionados.

A.3 Configuração dos Compute Nodes

À semelhança do *Head Node*, têm de se seguir alguns passos que se apresentam de seguida.

- Instalar o **CCP** no *Compute Node*.
  - *Join this Node to an existing Cluster* (Adiciona-se o *Compute Node* ao *Cluster* previamente criado).
    - Coloca-se o nome do *Head Node* para definir a associação ao *Cluster*.
  - Instalar o *Client Pack* para que os *Compute Nodes* possam submeter trabalhos.

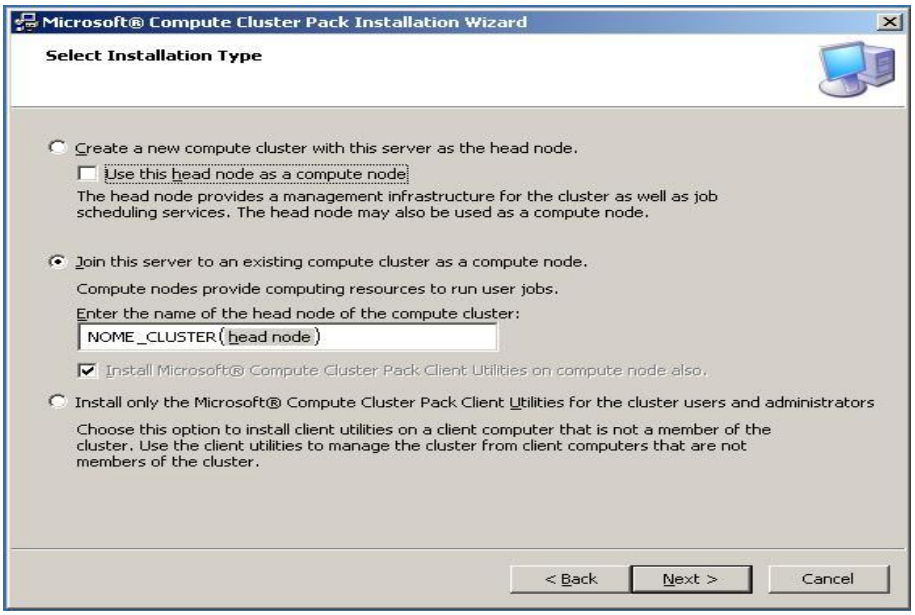

Figura A.6: Selecção de Head Node ou Compute Node.

Depois destas configurações feitas, adicionam-se então os *Compute Nodes* ao *Cluster*.

– Executa-se o *Compute Cluster Administrator Monitor*:

1. Adiciona-se o *Compute Node*

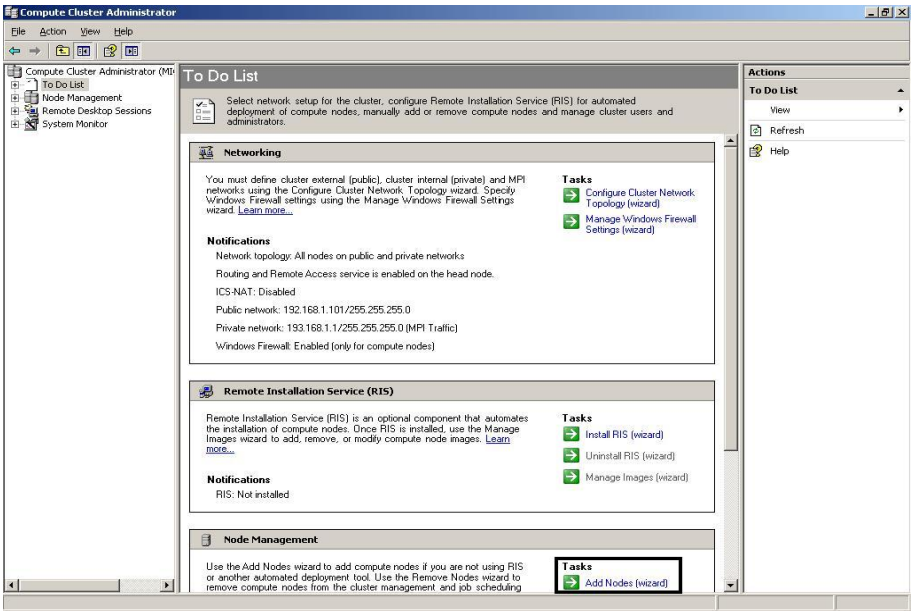

Figura A.7: Adição de um nó ao Cluster.

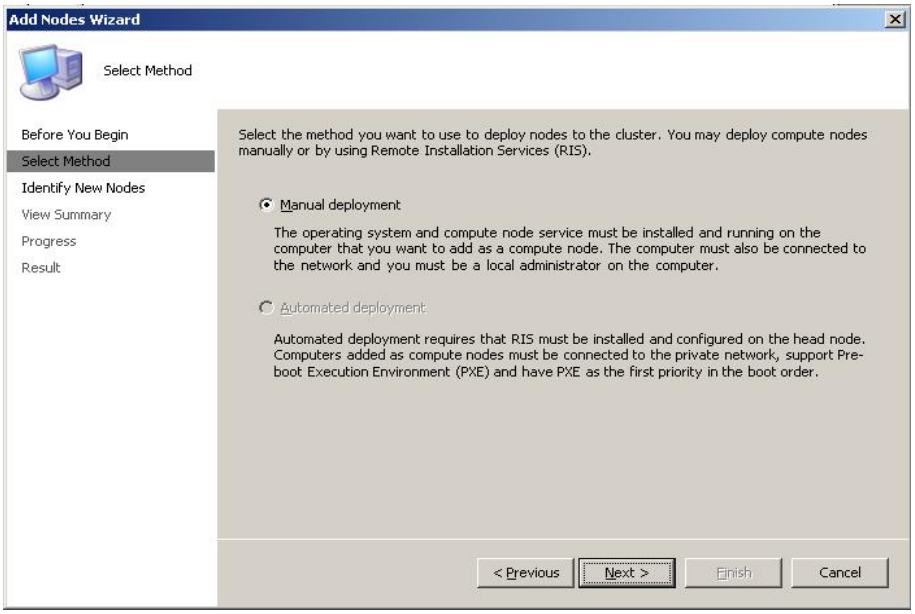

Figura A.8: Passo 2 da adição de um Nó.

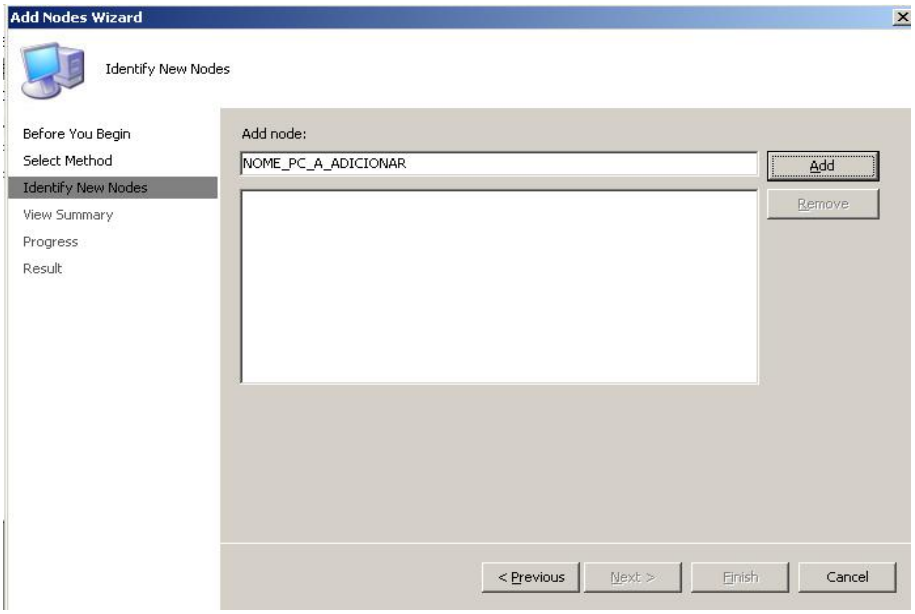

Figura A.9: Passo 3 da adição de um nó ao Cluster.

2. Aprova-se o PC no *Cluster*

3. Fazer *"Resume"* ao PC para que este esteja pronto a disponibilizar recursos de Computação.

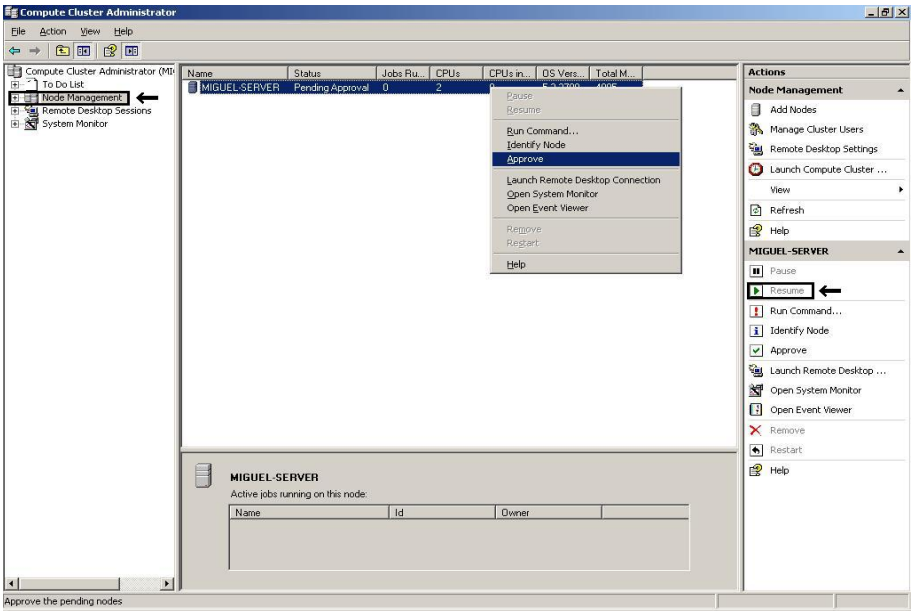

Figura A.10: Aprovação do Nó para utilização no Cluster.

Apêndice B

Configurações do SPW para funcionamento em Paralelo

Desde o início da criação deste software, já saíram várias versões. Neste momento já vai na sua versão 2.2.15 (ainda em modo experimental). Para a realização desta tese foi usada a versão 2.2.14.

B.1 SPW 2.2.14

B.1.1 Indicações para a instalação:

Dado que a versão 2.2.14 no momento apenas existe em formato .zip e não em formato .msi para instalação, a instalação da mesma é simples, basta apenas descompactar a versão para o mesmo directório onde está a versão anterior. Ex.: se temos a versão 2.2.13 instalada em D:/SPW2213, então a versão 2.2.14 estará também no D:/.
Assim sendo, o directório onde estarão os executáveis será D:/SPW2214Releasebuild.
Estas instruções são válidas apenas para quando se tem a versão anterior do programa instalada, pois já está também instalado o *ASPI Manager* e o *Sentinel*.

Caso não haja versões anteriores instaladas, o processo é similar. Descompactar o ficheiro .zip para a directoria pretendida, de seguida instalar *ASPI Manager* e o *Sentinel*. O *ASPI Manager* é instalado para que o Utilitário de I/O funcione correctamente. Se o *ASPI Manager* já estiver instalado, aparecerá a seguinte mensagem "*ASPI Layer is up to date!*". Caso contrário, será instalado.

O *Sentinel driver* é instalado para que o SO possa reconhecer o dispositivo de protecção, também conhecido por Chave (*Key, block, dongle*). Ao longo da instalação do *Sentinel*, dever-se-á aceitar a opção por defeito, para modificar a instalação já existente.

MUITO IMPORTANTE: ao seguir a instalação, num passo próprio, ter a certeza que se retiram todas as USB SuperPro Keys antes de continuar. Caso isto não seja feito, estas ficarão danificadas.

B.1.2 Configurações do SPW 2.2.14 para funcionar em modo paralelo:

Edit -► *Preferences*

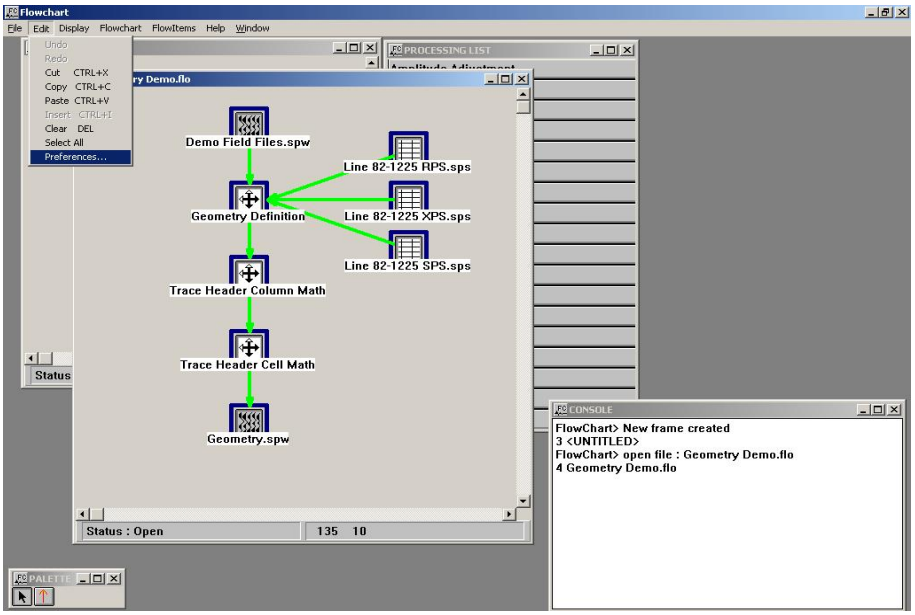

Figura B.1: Acesso ao Menu **Preferências**.

Alterar o modo de execução para "*Parallel*". Colocar o Host name (*Head Node do Cluster*), no caso **GOEFMAR15-GEO**.

De seguida colocar o *Target Directory for Compile files*, através do ***Path* de Rede**.

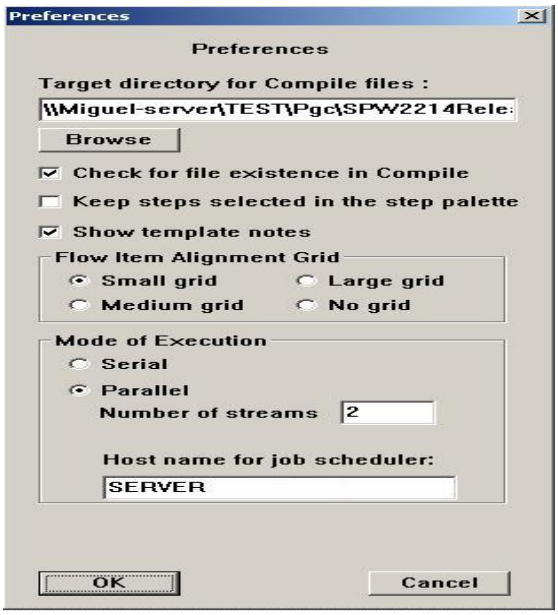

Figura B.2: Menu de Preferências.

Depois das configurações feitas, carregam-se os ficheiros a compilar. Estes deverão ser também carregados através do ***Path* de rede**. Os ficheiros de Dados **TÊM** todos que ser carregados com o ***Path* de Rede**.

Quando passamos a processar com dois ou mais nós do *cluster*, o *flow* não irá funcionar a não ser que a *Path Name* dos dados seja //... (*Network Name*). Se este nó permanecer pausado, então o *Job Scheduler* não irá tentar enviar-lhe tarefas. Se ele estiver activo, nas condições anteriores, pode ser-lhe enviada uma *Stream* de Execução (*Execution Stream*), o que levará a que o processo nunca seja terminado.

Quando se define o uso dois processadores, na verdade são alocados dois processadores mas apenas um é usado para processamento e o segundo para o controlo. Com quatro processadores, temos um para controlo e três processamento.

No fim das configurações resta apenas compilar usando o comando *Compile*.
Flowchart -¿ Compile

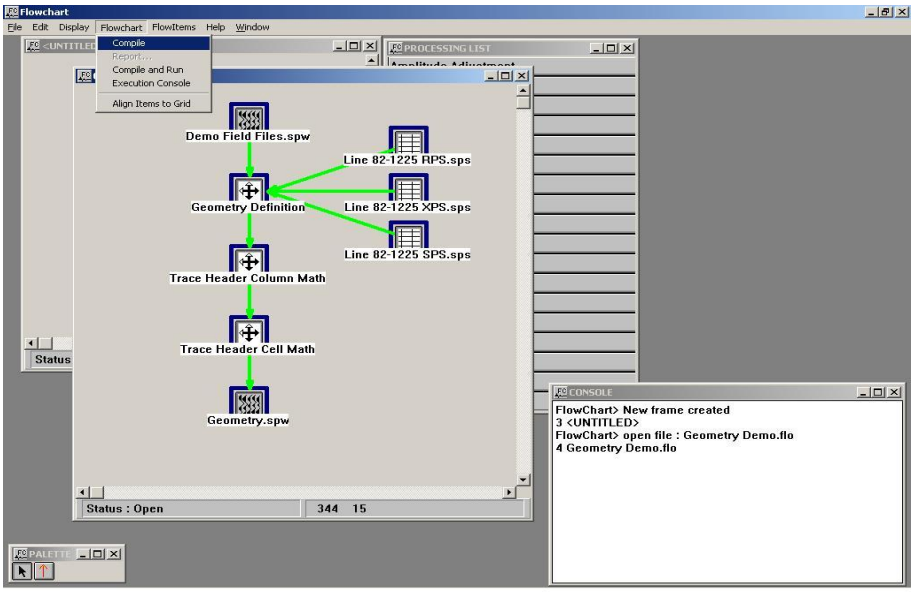

Figura B.3: Submissão de um trabalho para Execução no *Cluster*.

B.1.3 Notas Importantes

N.B.: De referir que é necessário criar um disco de rede partilhado, com permissões para que se possa escrever, ler, alterar, criar... Este disco será responsável por armazenar os ficheiros compilados. Caso as permissões não estejam correctamente atribuídas, o programa dá um erro e em muitos casos termina a execução.

N.B. 2: No início dos testes, quando a rede do *Cluster* não está ainda configurada, o facto de termos 2 nós adicionados ao *Cluster* (por exemplo: **Geofmar15-Geo** (*Head Node*) e **Geofmar17-Geo** (*Compute Node*)) pode causar uma situação de processamento infinito, dado que podem ser enviados dados para processar no Nó **Geofmar17-geo** e este não conseguirá retornar os resultados, dado que este não está correctamente configurado no *Cluster*. Desta forma, tem que pausar todos os nós e correr apenas processos no *Head Node*.

N.B. 3: Não é necessário que se corra o programa em *login* de Administrador. Deve-se ser capaz de o correr em qualquer tipo de *login* desde que este tenha as permissões correctas aplicadas à conta.

N.B. 4: De referir ainda que, em modo paralelo, nesta versão do SPW, a opção Compile and *Run* não funciona, pelo que não se consegue ver a Consola, com os registos. Algo que poderá ser corrigido numa versão posterior.

N.B. 5: Caso o *Job* não corra inicialmente, poderá ser necessário eliminar a dll: msmpi.dll

Bibliografia

[1]Akl, S. G. (1997). Parallel Computation, Models and Methods , Prentice Hall.

[2]Alfaro, J. C., Corcoran, C., Davies, K., Pineda, F. G., Hampson, G., Hill, D., Howhard, M., Kapoor, J., Moldeveanu, N. and Kragh, E. (2007). Reducing Exploration Risk, Oilfield Review.

[3]Corporation, M. (2009). "Acessando recursos entre florestas."Obtido em 29, Julho, 2009, de http://technet.microsoft.com/pt-br/library/cc772808(WS.10).aspx.

[4]Corporation, M. (2009). "Controladores de domínio."Obtido em 29, Julho, 2009, de http://technet.microsoft.com/pt-br/library/cc759623(WS.10).aspx.

[5]Corporation, M. (2009). "Direção da relação de confiança."Obtido em 29, Junho, 2009, de http://technet.microsoft.com/pt-br/library/cc728024(WS.10).aspx.

[6]Department of Oceanography, T. A. M. U. (2008). "Introduction to Physical Oceanography Course Schedule for Fall 2008."Obtido em 29, Setembro, 2009, de http://oceanworld.tamu.edu/ocean410/ocng410 *ourse chedule.html*.

[7]Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V. S. (1994). PVM Parallel Virtual Machine, The MIT Press.

[8]Gouveia, J. and Magalhães, A. (2005). Redes de Computadores , FCA.

[9]Ministério da Ciência, T. e. E. S. (2007) Cientistas Portugueses Participam no Desenvolvimento de Gigantesco Sistema de Computação GRID. 4

[10]Pereira, L. A. G. R. (2009). Atributos Sísmicos na Caracterização de Reservatórios de Hidrocarbonetos. Departamento de Geociências. Aveiro, Universidade de Aveiro. **Mestrado**: 165.

[11]Quin, M. J. (2003). Parallel Programming in C with MPI and Open MP , McGraw Hill.

[12]Sheriff, R. E. (1991). Encyclopedic dictionary of exploration geophysics . Tulsa, OK, Society of Exploration Geophysicists.

[13]Snir, M., Group, W., Huus-Lederman, S., Lumsdaine, A., Lusk, E., Nitzberg, B. and Saphir, W. (1998). MPI - The complete reference. Cambridge, Mass., MIT Press.

[14]Systems, F. (2008). "Cluster Beowulf."Obtido em 21, Setembro, 2009, de http://www.futureware.com.br/business/index.php/por/Solucoes/FW-Cluster/Cluster-Beowulf.

[15] Tanenbaum, A. S. (2006). Distributed Systems: Principles and Paradigms, Prentice Hall.

[16] Tománek, D. (2009). "Large-Scale Research Projects."Obtido em 21, Setembro, 2009, de http://www.pa.msu.edu/ tomanek/research.html.

[17] UCertify, E. (2006). "Windows Server 2003 Active Directory e Infra-estrutura de Rede."Obtido em 27, Julho, 2009, de http://www.wareprise.com/ja/2006/08/10/windows-server-2003-active-directory-and-network-infrastructure/pt/.

[18] University, M. T. (2004). "Computational Science Engineering Research Institute."Obtido em 21, Setembro, 2009, de http://www.cs.mtu.edu/ merk/Public/Websitedir/cseri.html.

[19] USGS (2009). "U.S. Geological Survey."Obtido em 21, Setembro, 2009, de http://woodshole.er.usgs.gov/openfile/of02-002/htmldocs/images/seismic.jpg.

[20] USGS (2009). "U.S. Geological Survey."Obtido em 21, Setembro, 2009, de http://woodshole.er.usgs.gov/.

[21] Wikipédia (2009). "Sismo."Obtido em 21, Julho, 2009, de http://pt.wikipedia.org/wiki/Sismo.

[22] Coulouris, G., Dollimore, J., Kindberg T. (2005). Distributed Systems - Concepts and Design, Addison-Wesley.